BCB546 - R Assignment

Mudith Ekanayake

3/19/2021

Part I

Data Inspection

Attributes of fang_et_al_genotypes.txt

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.4     v dplyr   1.0.4
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(tidyr)
```

To load the fang_et_al_genotypes.txt data file into R

```
fang_data = read_tsv("fang_et_al_genotypes.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   .default = col_character()
## )
## i Use `spec()` for the full column specifications.
```

To get the file size

```
file.size("fang_et_al_genotypes.txt")
```

```
## [1] 11051939
```

To get all the file info

```
file.info("fang_et_al_genotypes.txt", extra_cols = TRUE)
```

```
##                               size isdir mode               mtime
## fang_et_al_genotypes.txt 11051939 FALSE  666 2021-03-10 17:55:08
##                                        ctime               atime exe
## fang_et_al_genotypes.txt 2021-03-18 18:13:47 2021-03-24 23:39:28  no
```

To compactly display the internal structure of the R object

```
str(fang_data)
```

To get an idea about the data frame by viewing the first and last few rows

```
head(fang_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-aus~ TRIPS ?/?      ?/?      T/T   G/G   T/T   C/C   ?/?   G/G
## 2 SL-16     T-aus~ TRIPS ?/?      ?/?      T/T   ?/?   T/T   C/C   A/G   G/G
## 3 SL-11     T-bra~ TRIPS ?/?      ?/?      T/T   G/G   T/T   ?/?   G/G   G/G
## 4 SL-12     T-bra~ 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-dac~ TRIPS ?/?      ?/?      T/T   G/G   T/T   C/C   A/G   G/G
## # ... with 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>,
## #   PZA00029.11 <chr>, PZA00029.12 <chr>, PZA00030.11 <chr>, PZA00031.5 <chr>,
## #   PZA00041.3 <chr>, PZA00042.2 <chr>, PZA00042.5 <chr>, PZA00043.7 <chr>,
## #   PZA00045.1 <chr>, PZA00047.2 <chr>, PZA00049.12 <chr>, PZA00050.9 <chr>,
## #   PZA00051.2 <chr>, PZA00058.5 <chr>, PZA00058.6 <chr>, PZA00060.2 <chr>,
## #   PZA00061.1 <chr>, PZA00065.2 <chr>, PZA00069.4 <chr>, PZA00070.5 <chr>,
## #   PZA00078.2 <chr>, PZA00079.1 <chr>, PZA00081.17 <chr>, PZA00084.2 <chr>,
## #   PZA00084.3 <chr>, PZA00086.8 <chr>, PZA00088.3 <chr>, PZA00090.2 <chr>,
## #   PZA00092.1 <chr>, PZA00092.5 <chr>, PZA00093.2 <chr>, PZA00096.26 <chr>,
## #   PZA00097.13 <chr>, PZA00098.14 <chr>, PZA00100.10 <chr>, PZA00100.12 <chr>,
## #   PZA00100.14 <chr>, PZA00100.9 <chr>, PZA00103.20 <chr>, PZA00106.9 <chr>,
## #   PZA00107.18 <chr>, PZA00108.12 <chr>, PZA00108.14 <chr>, PZA00108.15 <chr>,
## #   PZA00109.3 <chr>, PZA00109.5 <chr>, PZA00111.2 <chr>, PZA00111.4 <chr>,
## #   PZA00111.5 <chr>, PZA00111.6 <chr>, PZA00111.8 <chr>, PZA00114.3 <chr>,
## #   PZA00116.2 <chr>, PZA00119.4 <chr>, PZA00120.4 <chr>, PZA00123.1 <chr>,
## #   PZA00125.2 <chr>, PZA00131.14 <chr>, PZA00132.17 <chr>, PZA00132.18 <chr>,
## #   PZA00132.3 <chr>, PZA00135.6 <chr>, PZA00137.2 <chr>, PZA00139.14 <chr>,
## #   PZA00140.10 <chr>, PZA00140.6 <chr>, PZA00140.9 <chr>, PZA00142.6 <chr>,
## #   PZA00148.2 <chr>, PZA00153.3 <chr>, PZA00153.6 <chr>, ...
```

```
tail(fang_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-I~ ZMMIL C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 2 S0398     Zmm-I~ ZMMIL G/G      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 3 S1636     Zmm-I~ ZMMIL G/G      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 4 CU0201    Zmm-I~ ZMMIL C/C      A/A      T/T   G/G   C/C   C/C   G/G   G/G
## 5 S0215     Zmm-I~ ZMMIL G/G      A/A      T/T   ?/?   C/C   C/C   G/G   G/G
## 6 CU0202    Zmm-I~ ZMMIL C/C      A/A      T/T   G/G   C/C   C/C   ?/?   G/G
## # ... with 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>,
## #   PZA00029.11 <chr>, PZA00029.12 <chr>, PZA00030.11 <chr>, PZA00031.5 <chr>,
## #   PZA00041.3 <chr>, PZA00042.2 <chr>, PZA00042.5 <chr>, PZA00043.7 <chr>,
## #   PZA00045.1 <chr>, PZA00047.2 <chr>, PZA00049.12 <chr>, PZA00050.9 <chr>,
## #   PZA00051.2 <chr>, PZA00058.5 <chr>, PZA00058.6 <chr>, PZA00060.2 <chr>,
## #   PZA00061.1 <chr>, PZA00065.2 <chr>, PZA00069.4 <chr>, PZA00070.5 <chr>,
## #   PZA00078.2 <chr>, PZA00079.1 <chr>, PZA00081.17 <chr>, PZA00084.2 <chr>,
## #   PZA00084.3 <chr>, PZA00086.8 <chr>, PZA00088.3 <chr>, PZA00090.2 <chr>,
## #   PZA00092.1 <chr>, PZA00092.5 <chr>, PZA00093.2 <chr>, PZA00096.26 <chr>,
## #   PZA00097.13 <chr>, PZA00098.14 <chr>, PZA00100.10 <chr>, PZA00100.12 <chr>,
## #   PZA00100.14 <chr>, PZA00100.9 <chr>, PZA00103.20 <chr>, PZA00106.9 <chr>,
## #   PZA00107.18 <chr>, PZA00108.12 <chr>, PZA00108.14 <chr>, PZA00108.15 <chr>,
## #   PZA00109.3 <chr>, PZA00109.5 <chr>, PZA00111.2 <chr>, PZA00111.4 <chr>,
## #   PZA00111.5 <chr>, PZA00111.6 <chr>, PZA00111.8 <chr>, PZA00114.3 <chr>,
## #   PZA00116.2 <chr>, PZA00119.4 <chr>, PZA00120.4 <chr>, PZA00123.1 <chr>,
## #   PZA00125.2 <chr>, PZA00131.14 <chr>, PZA00132.17 <chr>, PZA00132.18 <chr>,
## #   PZA00132.3 <chr>, PZA00135.6 <chr>, PZA00137.2 <chr>, PZA00139.14 <chr>,
## #   PZA00140.10 <chr>, PZA00140.6 <chr>, PZA00140.9 <chr>, PZA00142.6 <chr>,
## #   PZA00148.2 <chr>, PZA00153.3 <chr>, PZA00153.6 <chr>, ...
```

To get the dimensions of the data frame

```
dim(fang_data)
```

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

To get the number of rows in the data frame

```
nrow(fang_data)
```

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

To get the number of columns in the data frame

```
ncol(fang_data)
```

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

To get the structure of the data frame by previewing data in the columns

```
str(fang_data)
```

To view the column names

```
names(fang_data)
```

To see the class of all the columns

```
sapply(fang_data, class)
```

By inspecting this file I learned that:

- File size: 11051939 bytes
- Dimension of the dataframe: 2782 x 986
- Number of rows: 2782
- Number of columns: 986

Attributes of snp_position.txt

To load the snp_position.txt data file into R

```
snp_data = read_tsv("snp_position.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   SNP_ID = col_character(),
##   cdv_marker_id = col_double(),
##   Chromosome = col_character(),
##   Position = col_character(),
##   alt_pos = col_character(),
##   mult_positions = col_character(),
##   amplicon = col_character(),
##   cdv_map_feature.name = col_character(),
##   gene = col_character(),
##   `candidate/random` = col_character(),
##   Genaissance_daa_id = col_double(),
##   Sequenom_daa_id = col_double(),
##   count_amplicons = col_double(),
##   count_cmf = col_double(),
##   count_gene = col_double()
## )
```

To get the file size

```
file.size("snp_position.txt")
```

```
## [1] 82763
```

To get all the file info

```
file.info("snp_position.txt", extra_cols = TRUE)
```

```
##                   size isdir mode               mtime               ctime
## snp_position.txt 82763 FALSE  666 2021-03-10 17:55:08 2021-03-18 18:13:47
##                                atime exe
## snp_position.txt 2021-03-24 23:39:28  no
```

To compactly display the internal structure of the R object

```
str(snp_data)
```

```
## tibble [983 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ SNP_ID              : chr [1:983] "abph1.20" "abph1.22" "ae1.3" "ae1.4" ...
##  $ cdv_marker_id       : num [1:983] 5976 5978 6605 6606 6607 ...
##  $ Chromosome          : chr [1:983] "2" "2" "5" "5" ...
##  $ Position            : chr [1:983] "27403404" "27403892" "167889790" "167889682" ...
##  $ alt_pos             : chr [1:983] NA NA NA NA ...
##  $ mult_positions      : chr [1:983] NA NA NA NA ...
##  $ amplicon            : chr [1:983] "abph1" "abph1" "ae1" "ae1" ...
##  $ cdv_map_feature.name: chr [1:983] "AB042260" "AB042260" "ae1" "ae1" ...
##  $ gene                : chr [1:983] "abph1" "abph1" "ae1" "ae1" ...
##  $ candidate/random    : chr [1:983] "candidate" "candidate" "candidate" "candidate" ...
##  $ Genaissance_daa_id  : num [1:983] 8393 8394 8395 8396 8397 ...
##  $ Sequenom_daa_id     : num [1:983] 10474 10475 10477 10478 10479 ...
##  $ count_amplicons     : num [1:983] 1 0 1 0 0 1 1 0 1 0 ...
```

```
##  $ count_cmf           : num [1:983] 1 0 1 0 0 1 0 0 1 0 ...
##  $ count_gene          : num [1:983] 1 0 1 0 0 1 1 0 1 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   SNP_ID = col_character(),
##   ..   cdv_marker_id = col_double(),
##   ..   Chromosome = col_character(),
##   ..   Position = col_character(),
##   ..   alt_pos = col_character(),
##   ..   mult_positions = col_character(),
##   ..   amplicon = col_character(),
##   ..   cdv_map_feature.name = col_character(),
##   ..   gene = col_character(),
##   ..   `candidate/random` = col_character(),
##   ..   Genaissance_daa_id = col_double(),
##   ..   Sequenom_daa_id = col_double(),
##   ..   count_amplicons = col_double(),
##   ..   count_cmf = col_double(),
##   ..   count_gene = col_double()
##   .. )
```

To get an idea about the data frame by viewing the first and last few rows

```
head(snp_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~          5976 2          27403404 <NA>    <NA>           abph1
## 2 abph1~          5978 2          27403892 <NA>    <NA>           abph1
## 3 ae1.3           6605 5          1678897~ <NA>    <NA>           ae1
## 4 ae1.4           6606 5          1678896~ <NA>    <NA>           ae1
## 5 ae1.5           6607 5          1678898~ <NA>    <NA>           ae1
## 6 an1.4           5982 1          2404985~ <NA>    <NA>           an1
## # ... with 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>
```

```
tail(snp_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          2331285~ <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
## # ... with 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>
```

To get the dimensions of the data frame

```
dim(snp_data)
```

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

To get the number of rows in the data frame

```
nrow(snp_data)
```

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

To get the number of columns in the data frame

```
ncol(snp_data)
```

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

To get the structure of the data frame by previewing data in the columns

```
str(snp_data)
```

```
## tibble [983 x 15] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ SNP_ID              : chr [1:983] "abph1.20" "abph1.22" "ae1.3" "ae1.4" ...
##  $ cdv_marker_id       : num [1:983] 5976 5978 6605 6606 6607 ...
##  $ Chromosome          : chr [1:983] "2" "2" "5" "5" ...
##  $ Position            : chr [1:983] "27403404" "27403892" "167889790" "167889682" ...
##  $ alt_pos             : chr [1:983] NA NA NA NA ...
##  $ mult_positions      : chr [1:983] NA NA NA NA ...
##  $ amplicon            : chr [1:983] "abph1" "abph1" "ae1" "ae1" ...
##  $ cdv_map_feature.name: chr [1:983] "AB042260" "AB042260" "ae1" "ae1" ...
##  $ gene                : chr [1:983] "abph1" "abph1" "ae1" "ae1" ...
##  $ candidate/random    : chr [1:983] "candidate" "candidate" "candidate" "candidate" ...
##  $ Genaissance_daa_id  : num [1:983] 8393 8394 8395 8396 8397 ...
##  $ Sequenom_daa_id     : num [1:983] 10474 10475 10477 10478 10479 ...
##  $ count_amplicons     : num [1:983] 1 0 1 0 0 1 1 0 1 0 ...
##  $ count_cmf           : num [1:983] 1 0 1 0 0 1 0 0 1 0 ...
##  $ count_gene          : num [1:983] 1 0 1 0 0 1 1 0 1 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   SNP_ID = col_character(),
##   ..   cdv_marker_id = col_double(),
##   ..   Chromosome = col_character(),
##   ..   Position = col_character(),
##   ..   alt_pos = col_character(),
##   ..   mult_positions = col_character(),
##   ..   amplicon = col_character(),
##   ..   cdv_map_feature.name = col_character(),
##   ..   gene = col_character(),
##   ..   `candidate/random` = col_character(),
##   ..   Genaissance_daa_id = col_double(),
##   ..   Sequenom_daa_id = col_double(),
##   ..   count_amplicons = col_double(),
##   ..   count_cmf = col_double(),
##   ..   count_gene = col_double()
##   .. )
```

To view the column names

```
names(snp_data)
```

```
##  [1] "SNP_ID"              "cdv_marker_id"       "Chromosome"
##  [4] "Position"            "alt_pos"             "mult_positions"
```

```
##  [7] "amplicon"             "cdv_map_feature.name" "gene"
## [10] "candidate/random"     "Genaissance_daa_id"   "Sequenom_daa_id"
## [13] "count_amplicons"      "count_cmf"            "count_gene"
```

To see the class of all the columns

```
sapply(snp_data, class)
```

```
##               SNP_ID        cdv_marker_id           Chromosome
##          "character"            "numeric"          "character"
##             Position              alt_pos       mult_positions
##          "character"          "character"          "character"
##              amplicon cdv_map_feature.name                 gene
##          "character"          "character"          "character"
##     candidate/random   Genaissance_daa_id      Sequenom_daa_id
##          "character"            "numeric"            "numeric"
##      count_amplicons            count_cmf           count_gene
##            "numeric"            "numeric"            "numeric"
```

By inspecting this file I learned that:

- File size: 82763 bytes
- Dimension of the dataframe: 983 x 15
- Number of rows: 983
- Number of columns: 15

Data Processing

snp_data data frame was formatted such that the first column is "SNP_ID", the second column is "Chromosome", the third column is "Position".

```
snp_data <- snp_data[c(1,3,4)]
```

For maize

Filtered out maize data (Group = ZMMIL, ZMMLR, and ZMMMR) and "maize" data frame created.

```
maize <- fang_data %>% filter(Group=="ZMMIL"|Group=="ZMMLR"|Group=="ZMMMR")
```

Genotype data ("maize") were transposed using t() function so that the columns become rows. "stringsAsFactors = FALSE" prevents converting character columns to factors.

```
maize <- as.data.frame(t(maize), stringsAsFactors = FALSE)
```

rownames() function is the function that uses to get and set row names for data frames.

```
SNP_ID <- rownames(maize)
rownames(maize) <- NULL
```

cbind() function stands for column binding and it is normally used to combine vectors, matrices or data frames by columns. It splits matrix columns in data frame arguments and "stringsAsFactors = FALSE" prevents converting character columns to factors.

```
maize <- cbind(SNP_ID, maize, stringsAsFactors = FALSE)
```

First row was changed into SNP_ID and column 1, column 2, and column 3 were removed.

```
names(maize) <-  c("SNP_ID",maize[1,-1])
maize <- maize[-c(1,2,3), ]
```

Transposed maize genotype data and snp data were merged by SNP_ID.

```
merged_maize <- merge(snp_data, maize, by="SNP_ID")
```

A directory was Created for storing generated files for maize.

```
dir.create("maize_data")
```

mutate() function spcially can add new variables while preserving existing ones. arrange() function arranges rows by variables. Data were sorted based on position and were written to the ouputs as csv files. 10 files were generated (1 for each chromosome) with SNPs ordered based on increasing position values and with missing data.

```
for(i in c(1:10)){
  maize_data <- merged_maize %>% filter(Chromosome==i) %>% mutate(Position_new=as.numeric(Position)) %>%
  maize_data$Position_new <- NULL
  write.csv(maize_data, paste0("maize_chr_asc",i,".csv"), row.names = FALSE)
}
```

Data were sorted based on position and were written to the ouputs as csv files. 10 files were generated (1 for each chromosome) with SNPs ordered based on decreasing position values and with missing data.

```
for(i in c(1:10)){
  maize_data <- merged_maize %>% filter(Chromosome==i)%>% mutate(Position_new=as.numeric(Position)) %>%
  maize_data$Position_new <- NULL
  maize_data[maize_data == "?/?"] <- "-/-"
  write.csv(maize_data, paste0("maize_chr_dsc",i,".csv"), row.names = FALSE)
}
```

For Teosinte

Filtered out teosinte data (Group = ZMPBA, ZMPIL, and ZMPJA) and "teosinte" data frame created.

```
teosinte <- fang_data %>% filter(Group=="ZMPJA"|Group=="ZMPIL"|Group=="ZMPBA")
```

Genotype data ("teosinte") were transposed using t() function so that the columns become rows. "stringsAsFactors = FALSE" prevents converting character columns to factors.

```
teosinte <- as.data.frame(t(teosinte), stringsAsFactors = FALSE)
```

rownames() function is the function that uses to get and set row names for data frames.

```
SNP_ID <- rownames(teosinte)
rownames(teosinte) <- NULL
```

cbind() function stands for column binding and it is normally used to combine vectors, matrices or data frames by columns. It splits matrix columns in data frame arguments and "stringsAsFactors = FALSE" prevents converting character columns to factors.

```
teosinte <- cbind(SNP_ID, teosinte, stringsAsFactors = FALSE)
```

First row was changed into SNP_ID and column 1, column 2, and column 3 were removed.

```
names(teosinte) <-  c("SNP_ID", teosinte[1,-1])
teosinte <- teosinte[-c(1,2,3), ]
```

Transposed teosinte genotype data and snp data were merged by SNP_ID.

```
merged_teosinte <- merge(snp_data, teosinte, by="SNP_ID")
```

A directory was Created for storing generated files for teosinte.

```
dir.create("teosinte_data")
```

mutate() function spcially can add new variables while preserving existing ones. arrange() function arranges rows by variables. Data were sorted based on position and were written to the ouputs as csv files. 10 files were generated (1 for each chromosome) with SNPs ordered based on increasing position values and with missing data.

```
for(i in c(1:10)){
  teosinte_data <- merged_teosinte %>% filter(Chromosome==i) %>% mutate(Position_new=as.numeric(Positio
  teosinte_data$Position_new <- NULL
  write.csv(teosinte_data, paste0("teosinte_chr_asc",i,".csv"), row.names = FALSE)
}
```

Data were sorted based on position and were written to the ouputs as csv files. 10 files were generated (1 for each chromosome) with SNPs ordered based on decreasing position values and with missing data.

```
for(i in c(1:10)){
  teosinte_data <- merged_teosinte %>% filter(Chromosome==i)%>% mutate(Position_new=as.numeric(Position
  teosinte_data$Position_new <- NULL
  teosinte_data[teosinte_data == "?/?"]<-"-/-"
  write.csv(teosinte_data, paste0("teosinte_chr_dsc",i,".csv"), row.names = FALSE)
}
```

Part II

SNPs per chromosome

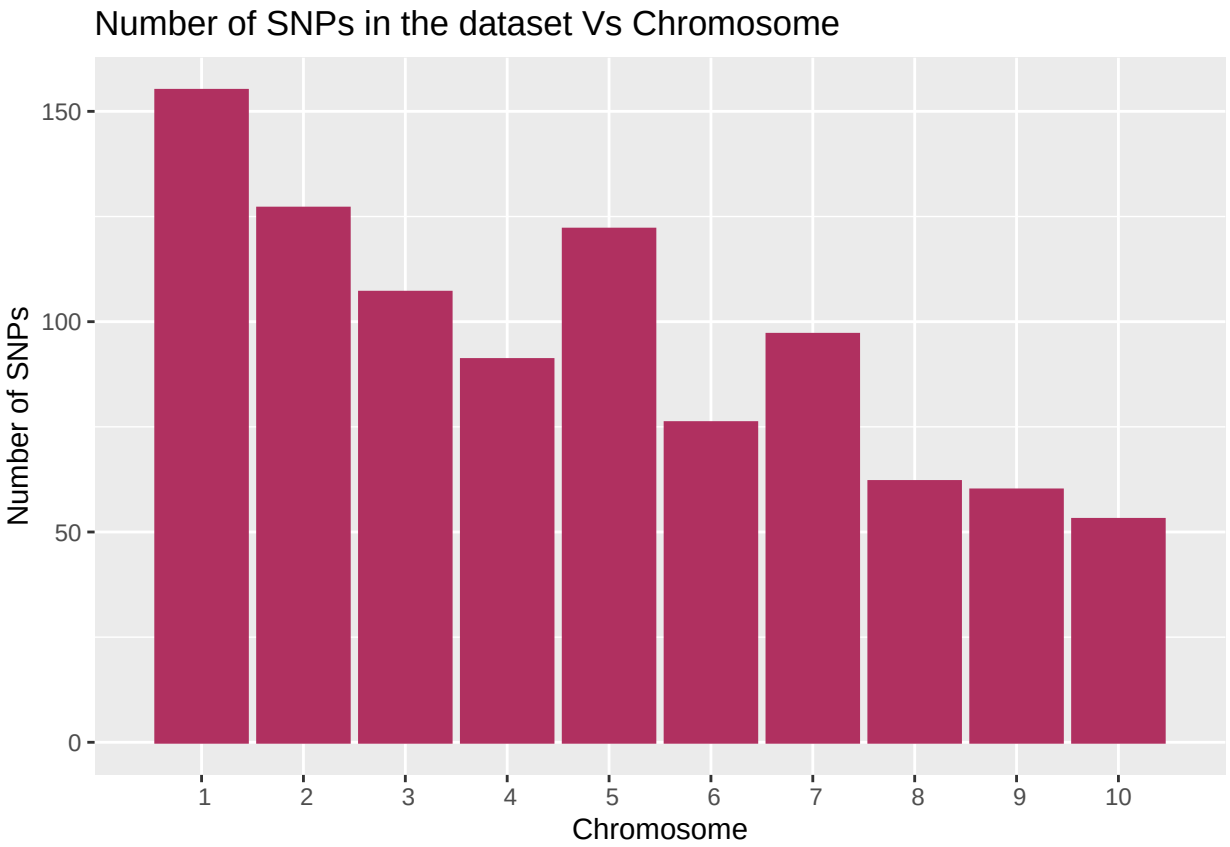
Plotting the total number of SNPs in the dataset on each chromosome.

is.numeric function checks whether its argument is numerical or not and is.na function checks whether there are missing values. "!" symbol reverses the function of is.na.

```
snp_data_n <- snp_data[!is.na(as.numeric(snp_data$Chromosome)),]
```

geom_bar() function was used to plot the bar chart. scale_x_discrete() function was used to set the values for discrete x aesthetic. labs() function was used to label the x and y axes of the graph. Bar chart was colored according to the "Chromosome". ggtitle() function was used to give a title to the graph.

```
ggplot(data = snp_data_n) + geom_bar(mapping = aes(as.numeric(Chromosome)), color = "maroon", fill = "ma
```



Plotting the distribution of SNPs on chromosomes.

Reshaping the original data using the pivot_longer() function in the tidyr package. Sample_ID, JG_OTU and Group columns were selected. "names_to" specifies the name of the column as "SNP_ID" and the column was created from the data stored in the column names of "fang_data" data frame. "values_to" specifies the name of the column as "NT" and the column was created from the data stored in cell values.

```
fang_pivot <- fang_data %>% pivot_longer(!c(Sample_ID, JG_OTU, Group), names_to="SNP_ID", values_to= "N
```

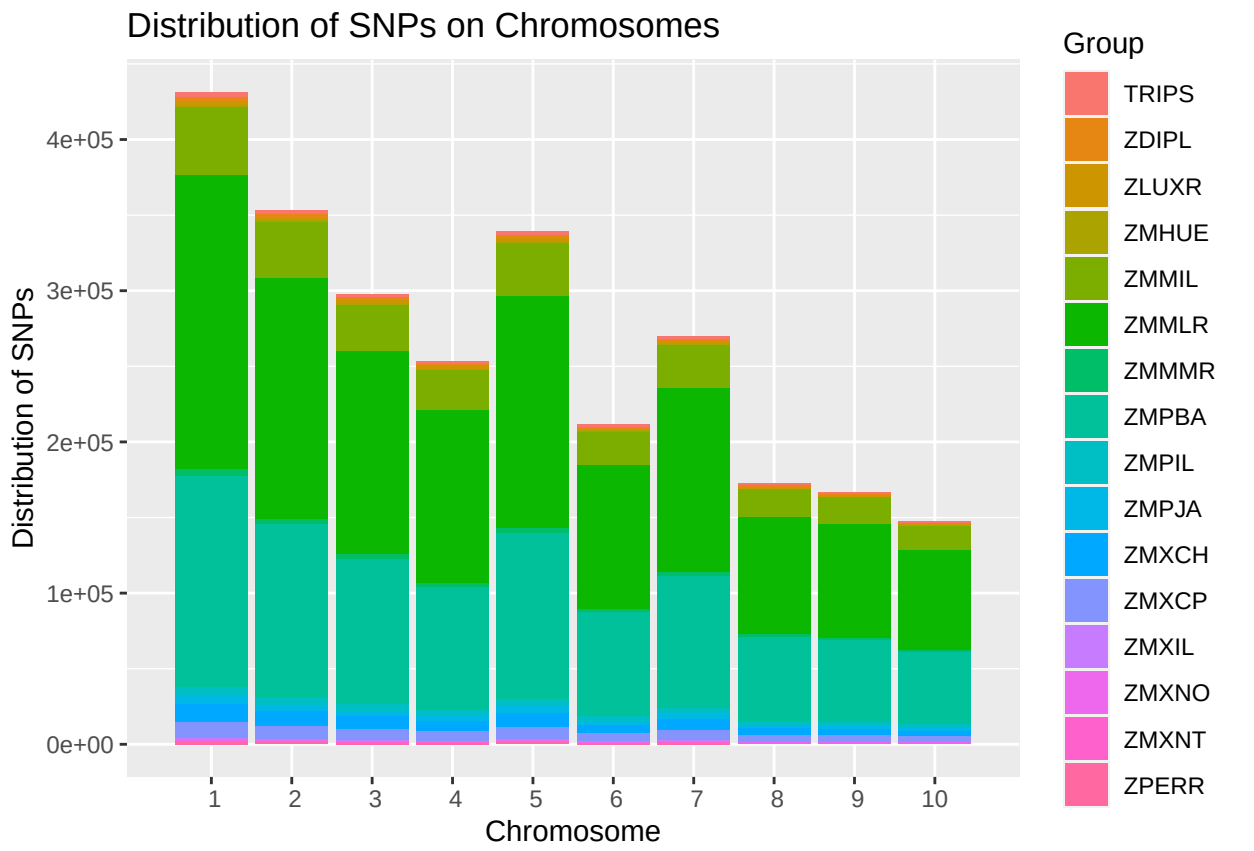
fang_pivot data frame was merged by SNP_ID.

```
merged_fang_Pivot <- merge(fang_pivot, snp_data, by="SNP_ID")
```

```
merged_fang_Pivot_n <- merged_fang_Pivot[!is.na(as.numeric(merged_fang_Pivot$Chromosome)),]
```

Bar chart was colored and shaded according to the "Group". x and y axes were labeled as Chromosome and Distribition of SNPs.

```
ggplot(data = merged_fang_Pivot_n) + geom_bar(mapping = aes(as.numeric(Chromosome), fill=Group)) + ggti
```



Missing data and amount of heterozygosity

Creating a new column named "homo_or_hetero" and indicating all the sites as "Heterozygous".

```
merged_fang_Pivot$homo_or_hetero <- "Heterozygous"
```

Check for the missing data and replace the sites with "Missing Data" in the "homo_or_hetero" column.

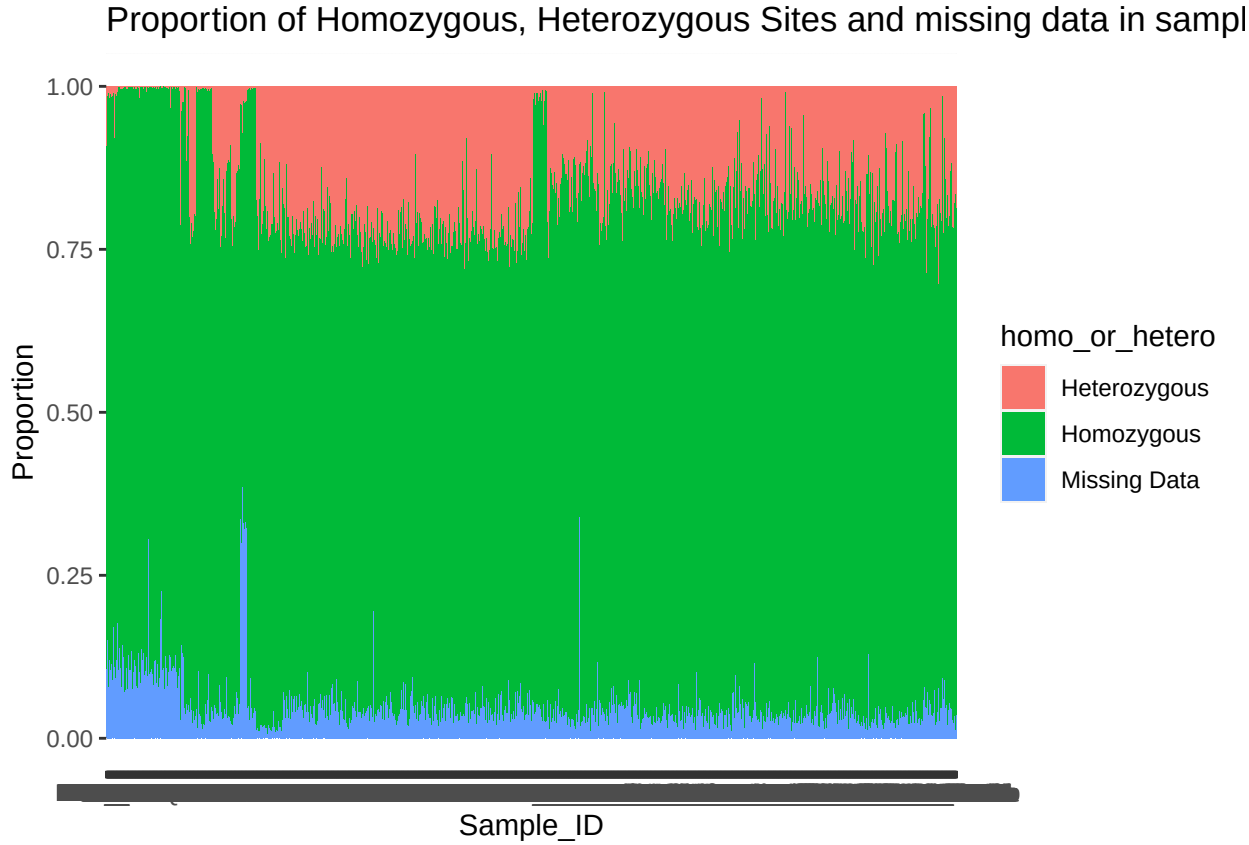
```
merged_fang_Pivot$homo_or_hetero[merged_fang_Pivot$NT == "?/?"] <- "Missing Data"
```

Check for the homozygous sites and replace the sites with "Homozygous" in the "homo_or_hetero" column.

```
merged_fang_Pivot$homo_or_hetero[merged_fang_Pivot$NT %in% c("A/A", "C/C", "G/G", "T/T")] <- "Homozygous
```

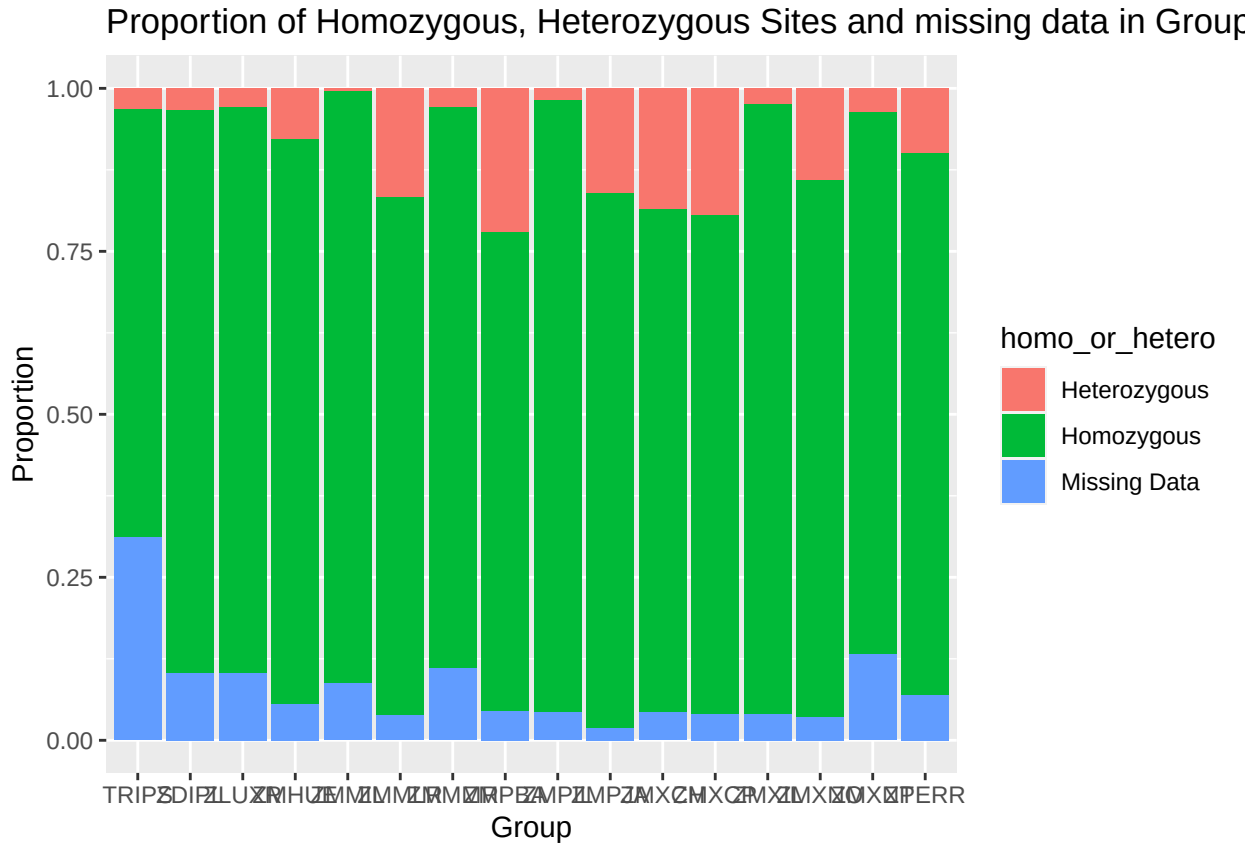
Proportion of homozygous and heterozygous sites as well as missing data in each sample was plotted using ggplot and height of the individual bars were normalized using ggplot's "position adjustment" option. Graph was labeled and titled using labs() and ggtitle() function.

```
ggplot(data = merged_fang_Pivot) + geom_bar(mapping=aes(x = Sample_ID, fill = homo_or_hetero), position
```



Proportion of homozygous and heterozygous sites as well as missing data in each Group was plotted using ggplot and height of the individual bars were normalized using ggplot's "position adjustment" option. Graph was labeled using labs() function.

```
ggplot(data = merged_fang_Pivot) + geom_bar(mapping = aes(x = Group, fill = homo_or_hetero), position =
```



My own visualization

Loading data again since previous data frame was manipulated.

```
snp_data_new = read_tsv("snp_position.txt")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   SNP_ID = col_character(),
##   cdv_marker_id = col_double(),
##   Chromosome = col_character(),
##   Position = col_character(),
##   alt_pos = col_character(),
##   mult_positions = col_character(),
##   amplicon = col_character(),
##   cdv_map_feature.name = col_character(),
##   gene = col_character(),
##   `candidate/random` = col_character(),
##   Genaissance_daa_id = col_double(),
##   Sequenom_daa_id = col_double(),
##   count_amplicons = col_double(),
##   count_cmf = col_double(),
##   count_gene = col_double()
## )
```

New data frame was created selecting only the Chromosome column and gene column.

```
gene_distribution <- snp_data_new %>% select(Chromosome, gene)
```

Duplicate rows which match same gene for same chromosome were removed in order to make the data frame simple for counting genes.

```
deduped.data <- unique( gene_distribution[ , 1:2 ] )
```

Genes per chromosome were counted using count() function.

```
gene_count <- count(deduped.data, Chromosome)
```

Bar plot was generated using ggplot(). "stat='identity' " was included since y values were calculated in the previous step and they were provided separately in the aes() function.

```
ggplot(gene_count, aes(x = Chromosome, y = n, fill = Chromosome)) + geom_bar(stat='identity') + ggtitle
```