R_assignment_FINAL

2025-03-14

Replication of UNIX assignment in R

```
# libraries
library(tidyverse) # basic R functions
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
## -- 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
```

```
library(janitor) # df clean up
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(tibble) # df clean up
library(tidyr) # df clean up
library(ggplot2) # data visualization
library(patchwork) # data visualization

# NOTE: In case git commit & push are too large, use: `git config --global http.postBuffer 1048576000`

# read-in (customize to your own computer director: "/.../.../raw_data/fang_et_al_genotypes.txt")
genotype_path <- "/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R/raw_data/fang_et
snp_path <- "/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R/raw_data/snp_positio

genotypes <- read_table(genotype_path)
```

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

```
snp <- read_table(snp_path)
```

```
##
## -- 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_character(),
##   Sequenom_daa_id = col_character(),
##   count_amplicons = col_character(),
##   count_cmf = col_character(),
##   count_gene = col_character()
## )
```

```
## Warning: 983 parsing failures.
## row col   expected     actual
##   1  -- 15 columns 13 columns '/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R_
##   2  -- 15 columns 13 columns '/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R_
##   3  -- 15 columns 13 columns '/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R_
##   4  -- 15 columns 13 columns '/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R_
##   5  -- 15 columns 13 columns '/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R_
## ... ... .......... .......... ..........................................................................
## See problems(...) for more details.
```

```r
# dimensions (# of rows & # of columns)
dim(genotypes)
dim(snp)

# variable names
colnames(genotypes)
colnames(snp)

# variable data type
unlist(lapply(genotypes, class))
unlist(lapply(snp, class))

# str(genotypes)    # can remove "#" from this code if desired; removed for storage when knitting
# str(snp)

# inital look (first 6 rows of each df)
head(genotypes)
head(snp)
```

Data Inspection Explanation

- The dimensions of the `fang_et_al_genotypes` file is 2782 observations (rows) with 986 variables (columns)
- The dimensions of the `snp_position` file is 983 observations (rows) with 15 variables (columns)

- First look at the variable names of the `fang` file displays "Sample_ID", "JG_OTU", and "Group" followed by a list of markers
- First look at the variable names of the `snp` file displays the key variables needed in this analysis ("SNP_ID", "Chromosome", and "Position") along with 12 other variables irrelevant to this particular analysis
- For both the `fang` and `snp` files, all the variables are "character" type, which may pose issues in the future with ordering
- The head function displays the first 10 rows of each file revealing the variables in the `fang` file match the marker IDs in the "SNP_ID" column in the `snp` file
  - This also gives insight on the content of the data with the `fang` file including mostly genotype calls at SNP markers using biallelic notation (allele/allele), and the `snp`file including information about the SNP markers, such as the chromosome(s) they are located on and where on the chromosome they are located (described by the base pair number)

Data Processing Filtering

The following section filters the original `fang` file into two data frames (one for maize & one for teosinte) and removes unnecessary variables. The `all_other_groups` ensures that filtering worked as the observations for all three dfs should add up to the original number of observations in genotypes. This also prepares the `snp` file by removing unnecessary variables for this analysis.

```
# filtering for maize (Group = ZMMIL, ZMMLR, and ZMMMR) & remove JG_OTU and Group column
maize <- genotypes |>
  filter(Group %in% c("ZMMIL", "ZMMLR", "ZMMMR")) |>
  select(!c(Group, JG_OTU))

# filtering for teosinte (Group = ZMPBA, ZMPIL, and ZMPJA)  remove JG_OTU and Group column
teosinte <- genotypes |>
  filter(Group %in% c("ZMPBA", "ZMPIL", "ZMPJA")) |>
  select(!c(Group, JG_OTU))

# checking filtering success
all_other_groups <- genotypes |>
  filter(!Group %in% c("ZMMIL", "ZMMLR", "ZMMMR", "ZMPBA", "ZMPIL", "ZMPJA"))
total_obs <- nrow(all_other_groups) + nrow(maize) + nrow(teosinte)
total_obs #2782 checks out

# select for "SNP_ID", "Chromosome", and "Position" columns in `snp` file
snp_filt <- snp |>
  select(c("SNP_ID", "Chromosome", "Position"))
```

Transposing, Joining

The following section transposes the `maize` and `teosinte` data frames, ensures the column names are accurate, and creates a column labeled "SNP_ID" containing the SNP markers that match the SNP IDs in the `snp_filt` df to prepare for joining. The two genotype files are joined with the `snp_filt` df to add the chromosome and snp position into the df. Since the columns are added at the end of the column names, they are relocated to the beginning for easy examination.

```
# ~~~ MAIZE ~~~

# transposed maize df
maize_t <- as.data.frame(t(maize))

# row 1 to column names & row names to column
maize_t <- maize_t |>
```

```
  row_to_names(row_number = 1) |>
  rownames_to_column("SNP_ID")

# joining `snp` & `maize_t` by "SNP_ID"
maize_join <- full_join(maize_t, snp_filt, by = "SNP_ID")

# moving "Chromosome" & "Position" columns forward
maize_join <- maize_join |>
  relocate(ncol(maize_join)-1, ncol(maize_join), .after = 1)
```

```
# ~~~ TEOSINTE ~~~

# transposed teosinte df
teosinte_t <- as.data.frame(t(teosinte))

# row 1 to column names & row names to column
teosinte_t <- teosinte_t |>
  row_to_names(row_number = 1) |>
  rownames_to_column("SNP_ID")

# joining `snp` & `teosinte_t` by "SNP_ID"
teosinte_join <- full_join(teosinte_t, snp_filt, by = "SNP_ID")

# moving "Chromosome" & "Position" columns forward
teosinte_join <- teosinte_join |>
  relocate(ncol(teosinte_join)-1, ncol(teosinte_join), .after = 1)
```

Filtering for new file output

To filter, edit, and create output files efficiently, I designed a function that completes all steps required when provided the following input information:

- *date* = the input df (`maize_join` or `teosinte_join`)
- *chr_num* = the chromosome number to filter for (`1-10`)
- *output_prefix* = prefix for naming the output file (`maize` or `teosinte`)
- *replace_na* = the string to replace "?/?" values (`?/?` or `-/-`)
- *sort_order* (default = "asc") = specifies whether to sort position by ascending ("asc") or descending ("desc") order; if no input is added, the function will automatically sort position in ascending order
- *output_dir* = directory (aka folder) output files are saved in

To ensure the function works properly for each file output type & show how the function is used, the full code for chromosome 1 (ascending & descending) for both maize and teosinte are typed. The files output for chromosomes 2-10 were created using a lapply function, which applies a function (`process_chr_data`) to chromosomes (`2:10`).

```
# ~~~ FUNCTION CREATION ~~~

process_chr_data <- function(data, chr_num, output_prefix, replace_na, sort_order = "asc", output_dir =
  # create file output directory (unless it already exists)
  if(!dir.exists(output_dir)) {
    dir.create(output_dir, recursive = FALSE)
  }

  # filter data
  filtered_data <- data |>
    filter(Chromosome == chr_num, !grepl("unknown|multiple", Chromosome)) |>
```

```
    mutate(
      across(everything(), ~ gsub("\\?/\\?", replace_na, .)),
      Position = as.numeric(as.character(Position))
      ) |>
    arrange(if (sort_order == "asc") Position else desc(Position))

  # write to file
  output_file <- file.path(output_dir, sprintf("%s_chr%d_%s.txt", output_prefix, chr_num, sort_order))
  write_tsv(filtered_data, output_file)
}
```

```
# ~~~ OUTPUT FILE CREATION ~~~

# ~~~ maize ascending ~~~
process_chr_data(
  data = maize_join,
  chr_num = 1, # chromosome 1
  output_prefix = "maize",
  replace_na = "?/?",
  sort_order = "asc",
  output_dir = "maize_data")

lapply(2:10, function(chr) { # chromosomes 2 - 10
  process_chr_data(maize_join, chr, "maize", "?/?", "asc", "maize_data")
})
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
# ~~~ maize descending ~~~
process_chr_data(
  data = maize_join,
  chr_num = 1, # chromosome 1
  output_prefix = "maize",
  replace_na = "-/-",
  sort_order = "desc",
  output_dir = "maize_data")
```

```
lapply(2:10, function(chr) { # chromosomes 2 - 10
  process_chr_data(maize_join, chr, "maize", "-/-", "desc", "maize_data")
})
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
# ~~~ teosinte ascending ~~~
process_chr_data(
  data = teosinte_join,
  chr_num = 1, # chromosome 1
  output_prefix = "teosinte",
  replace_na = "?/?",
  sort_order = "asc",
  output_dir = "teosinte_data")

lapply(2:10, function(chr) { # chromosomes 2 - 10
  process_chr_data(teosinte_join, chr, "teosinte", "?/?", "asc", "teosinte_data")
})
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
```

```
# ~~~ teosinte descending ~~~
process_chr_data(
  data = teosinte_join,
  chr_num = 1, # chromosome 1
  output_prefix = "teosinte",
  replace_na = "-/-",
  sort_order = "desc",
  output_dir = "teosinte_data")

lapply(2:10, function(chr) { # chromosomes 2 - 10
  process_chr_data(teosinte_join, chr, "teosinte", "-/-", "desc", "teosinte_data")
})
```

```
## Warning: There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
## There was 1 warning in `mutate()`.
## i In argument: `Position = as.numeric(as.character(Position))`.
## Caused by warning:
## ! NAs introduced by coercion
```

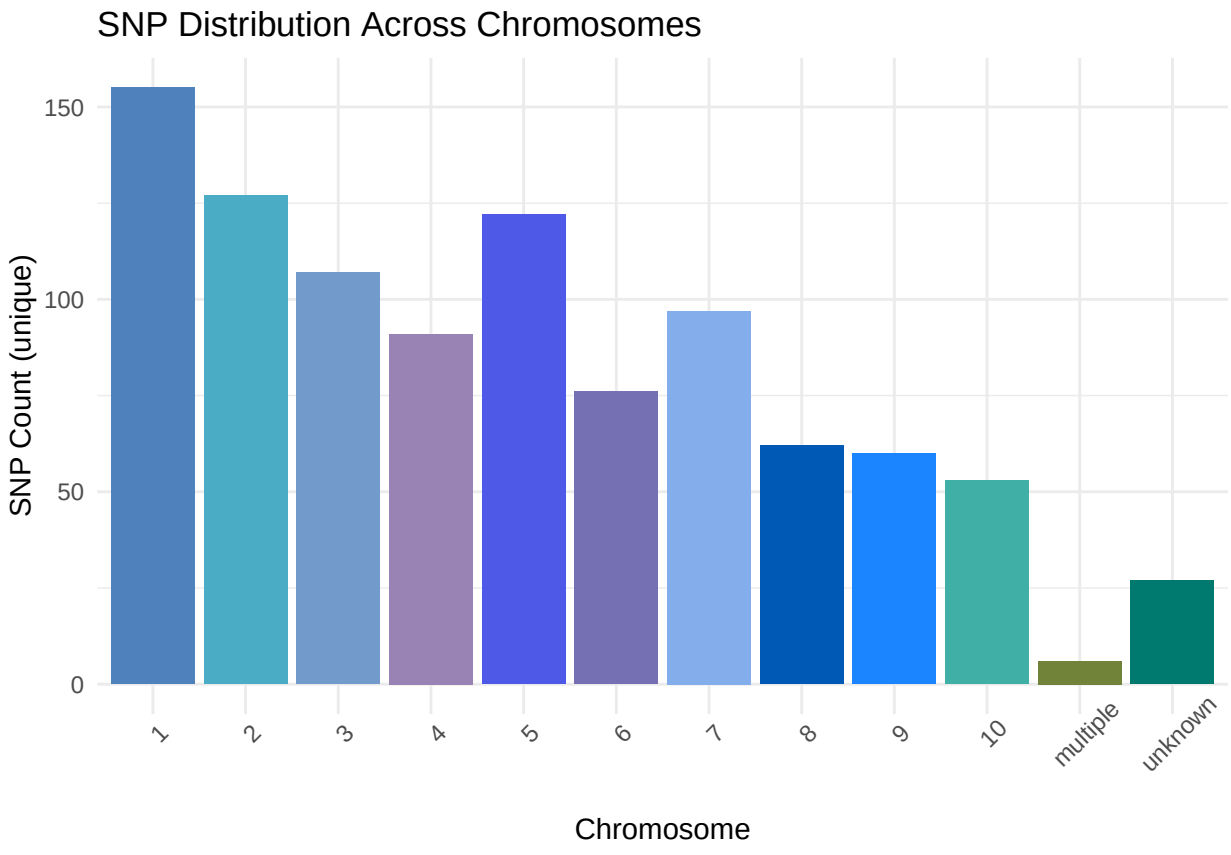
Data Visualization

Distribution of SNPs on and across chromosomes

```
# ~~~ distribution of SNPs on chromosomes (how many SNPs on each chromosome) ~~~

# join maize_join & teosinte_join
snp_plot_df <- full_join(maize_join, teosinte_join, by = c("SNP_ID", "Chromosome", "Position"))

# ensure chromosomes are in correct order
chromosome_order <- c(as.character(1:10), "multiple", "unknown")
snp_plot_df <- snp_plot_df |>
  mutate(Chromosome = factor(Chromosome, levels = chromosome_order))
```

```
# distribution of SNP per chromosome
plot1a <- ggplot(snp_plot_df, aes(x = Chromosome, fill = Chromosome)) +
  geom_bar() +
```

```
  labs(
    title = "SNP Distribution Across Chromosomes",
    x = "Chromosome",
    y = "SNP Count (unique)"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45),
    legend.position = "none") +
  scale_fill_manual(
    name = "Chromosome",
    values = c(
      "#4f81bd", "#4bacc6", "#729aca", "#9983b5", "#4e5ae7", "#7470b3", "#83adeb", "#005AB5", "#1A85FF"
"#007a6e"
    )
  )

plot1a
```




```
ggsave("SNP Distribution Across Chromosomes.png", plot = plot1a)
```

```
## Saving 6.5 x 4.5 in image
```

Explanation

- First, the separated `maize_join` and `teosinte_join` dfs were rejoined by the "SNP_ID", "Chromosome", and "Position" columns. A chromosome order was established and assigned to the df to ensure correct order when plotting data.

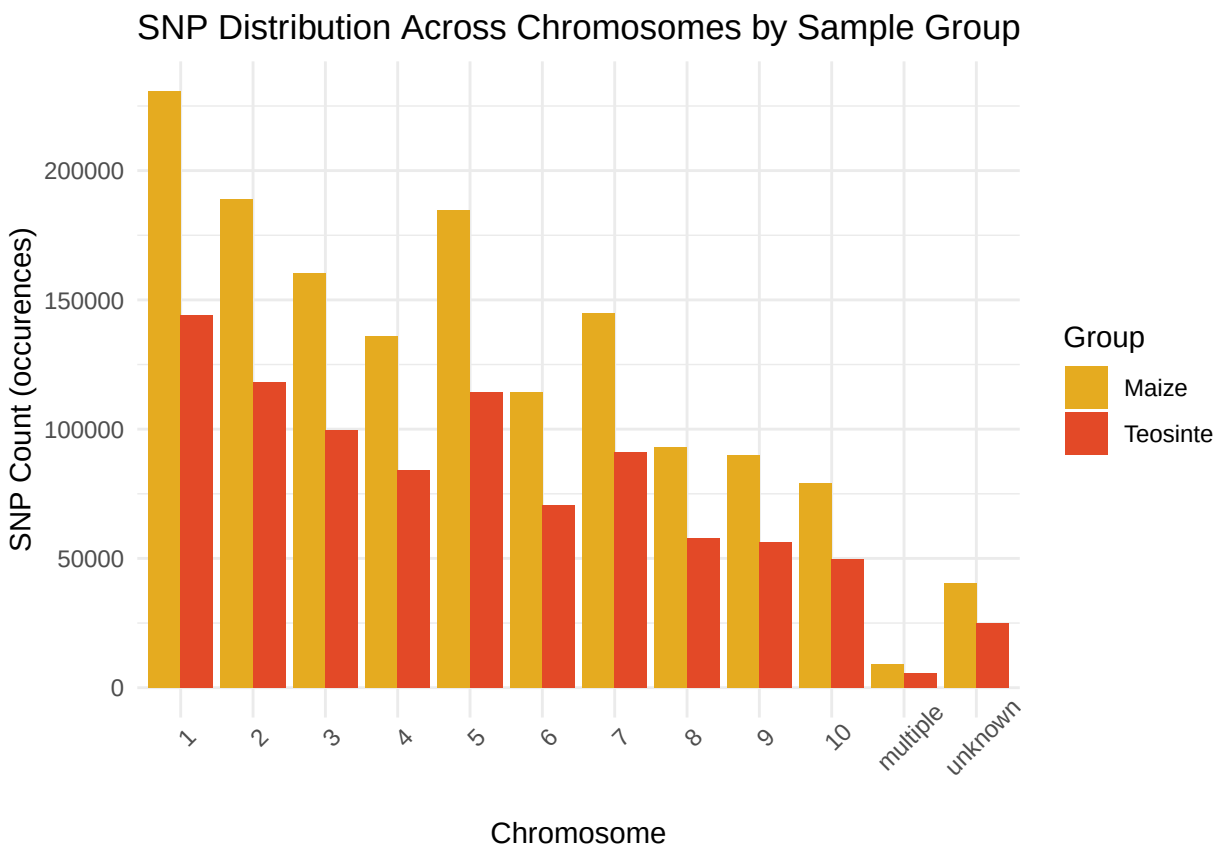
- The plot displays the counts of unique SNP ids for each chromosome, including SNPs on multiple or unknown chromosomes.
- Chromosome 1 has the most unique SNPs present with Chromosome 5 and 7 having the second and third most, respectfully. Other than those three chromosomes, the SNP count decreases as the chromosome number increases.
- There are few SNPs located on multiple chromosomes.

```
# ~~~ distribution of SNP occurrences on chromosomes (SNP occurrences across chromosomes in maize and t

# read-in "snp_plot2_df" file from "working_files" folder on github repository
# look at "snp_plot_data_prep.Rmd" on git hub repository for how this file was created
snp_plot2_df_path <- "/Users/lilliang/Documents/Spring_2025/EEOB5460/R_assignment/EEOB546_R/working_fil

snp_plot2_df <- read_table(snp_plot2_df_path)
```

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

```
# ensure Chromosome's are in correct order
chromosome_order <- c(as.character(1:10), "multiple", "unknown")
snp_plot2_df <- snp_plot2_df |>
  mutate(Chromosome = factor(Chromosome, levels = chromosome_order))

# reshape data to long format and filter non-`"?/?"` SNPs
snp_long <- snp_plot2_df |>
  pivot_longer(cols = c(contains("_m"), contains("_t")), names_to = "Sample", values_to = "Genotype") |
  filter(Genotype != "?/?") |>
  mutate(
    Chromosome = factor(Chromosome),  # ensure Chromosome is a factor
    Group = ifelse(grepl("_m", Sample), "Maize", "Teosinte") # assign group
  )
```

```
plot1b <- ggplot(snp_long, aes(x = Chromosome, fill = Group)) +
  geom_bar(position = "dodge") +
  labs(title = "SNP Distribution Across Chromosomes by Sample Group",
       x = "Chromosome",
       y = "SNP Count (occurences)") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45)) +
  scale_fill_manual(name = "Group",
                    values = c("#e5ab20", "#e34927"))

plot1b
```




```
ggsave("SNP Distribution Across Chromosomes by Sample Group.png", plot = plot1b)
```

```
## Saving 6.5 x 4.5 in image
```

Explanation

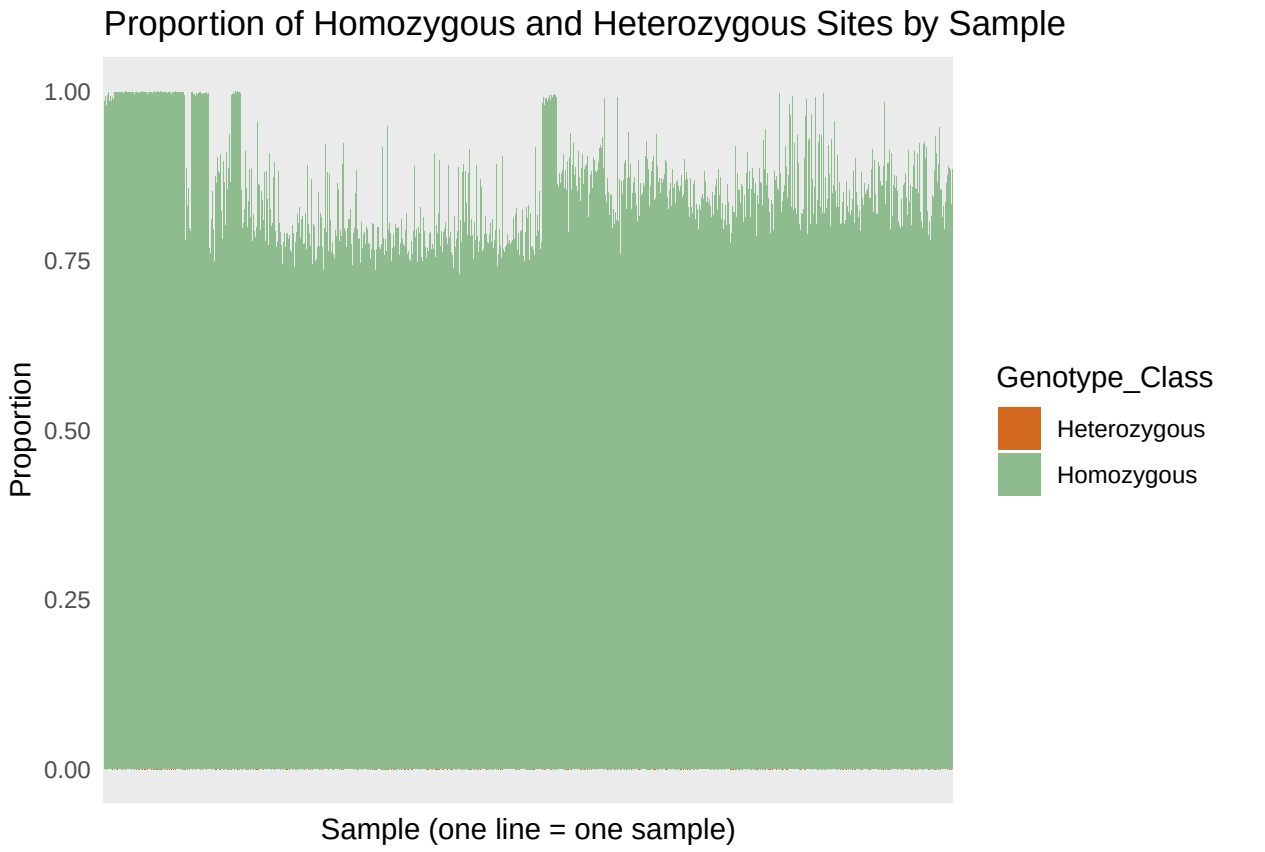
- An additional dataset must be read in from the working_files folder. The code used to generate this dataset is available in the GitHub repository under snp_plot_data_prep.Rmd.
- To distinguish between groups in the data, an identifier was added to the sample names, facilitating better filtering when pivoting the table into a long format.
- Samples with unknown nucleotides were removed, as they will be analyzed separately later.
- The plot visualizes the total number of times SNPs appears in a sample per chromosome. The two groups, maize and teosinte, are color-coded to highlight differences in SNP distributions.
- The observed trends align with the previous plot ("SNP Distribution Across Chromosomes"): Chromosomes 1, 5, and 7 exhibit the highest SNP occurrences, which is expected given the greater number of SNPs present on these chromosomes.
- SNP counts are higher in maize samples, though this may be attributed to the larger number of maize samples collected (1,573 vs. 975), which could explain the visual difference in counts.

Missing data & amount of heterozygosity

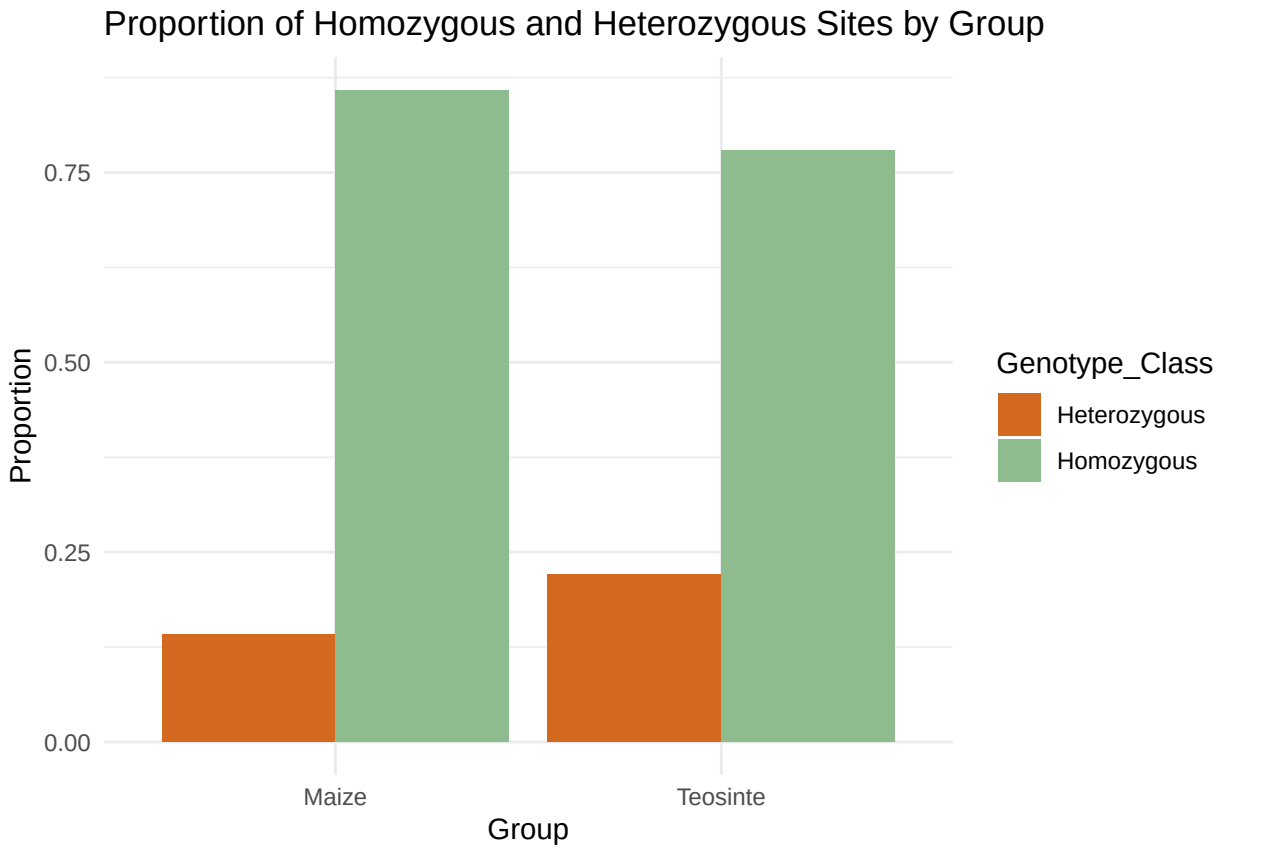
```
# ~~~ HETEROZYGOSITY ~~~

# create column that differentiates hom. vs. het. genotypes
snp_long$Genotype_Class <- ifelse(snp_long$Genotype %in% c("A/A", "C/C", "G/G", "T/T"), "Homozygous", "
# proportion of homozygous and heterozygous sites for each sample
hom_prop_sample <- table(snp_long$Sample, snp_long$Genotype_Class)
hom_prop_sample <- prop.table(hom_prop_sample, 1)
```

```r
# proportion of homozygous and heterozygous sites for each group
hom_prop_group <- table(snp_long$Group, snp_long$Genotype_Class)
hom_prop_group <- prop.table(hom_prop_group, 1)

# convert proportion tables to data frames for plotting
hh_prop_sample_df <- as.data.frame(as.table(hom_prop_sample))
hh_prop_group_df <- as.data.frame(as.table(hom_prop_group))

colnames(hh_prop_sample_df) <- c("Sample", "Genotype_Class", "Proportion")
colnames(hh_prop_group_df) <- c("Group", "Genotype_Class", "Proportion")
```

```r
# plot for each Sample
plot2a <- ggplot(hh_prop_sample_df, aes(x = Sample, y = Proportion, fill = Genotype_Class)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Proportion of Homozygous and Heterozygous Sites by Sample",
    x = "Sample (one line = one sample)",
    y = "Proportion") +
  scale_fill_manual(values = c("Homozygous" = "darkseagreen", "Heterozygous" = "chocolate")) +
  theme_minimal() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())

# plot for each Group
plot2b <- ggplot(hh_prop_group_df, aes(x = Group, y = Proportion, fill = Genotype_Class)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Proportion of Homozygous and Heterozygous Sites by Group",
    x = "Group",
    y = "Proportion") +
  scale_fill_manual(values = c("Homozygous" = "darkseagreen", "Heterozygous" = "chocolate")) +
  theme_minimal()

plot2a
```


Proportion of Homozygous and Heterozygous Sites by Sample
1.00
0.75
0.50
0.25
0.00
Proportion
Sample (one line = one sample)
Genotype_Class
Heterozygous
Homozygous

```
plot2b
```




```
ggsave("Proportion of Homozygous and Heterozygous Sites by Sample.png", plot = plot2a)
```

```
## Saving 6.5 x 4.5 in image
```

```
ggsave("Proportion of Homozygous and Heterozygous Sites by Group.png", plot = plot2b)
```

```
## Saving 6.5 x 4.5 in image
```

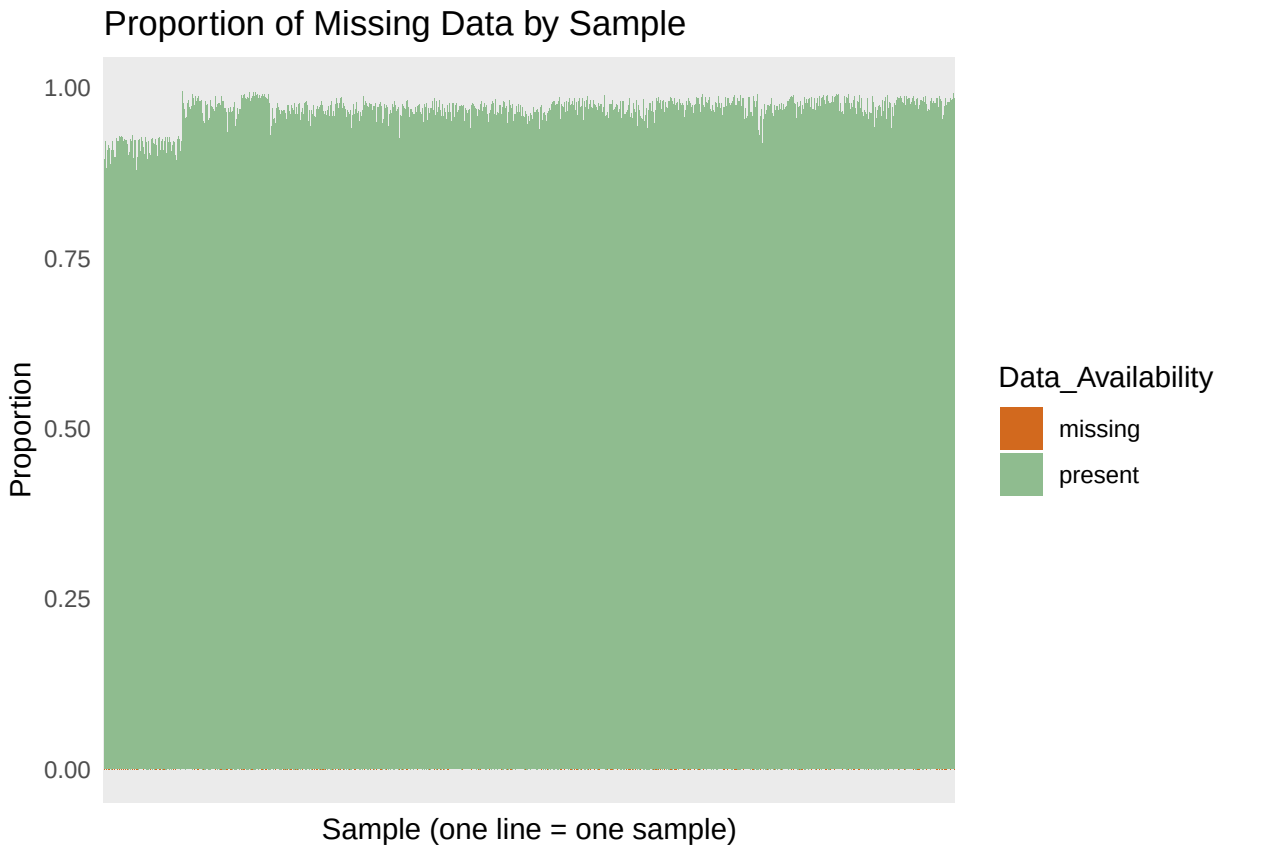
Explanation

- An additional column in `snp_long` df was generated separating the homozygous and heterozygous genotypes. The proportion of homozygous and heterozygous sites for each sample and each group were calculated and plotted.
- Plot 2a displays the proportion of homozygous and heterozygous sites for each sample (one line = one sample); however this figure is hard to read due to the high number of samples.
- Plot 2b groups the data by species (maize & teosinte) and displays the proportion of homozygous and heterozygous sites in a much simpler way. In this plot, it is obvious that there are significantly more homozygous genotypes than heterozygous in both the maize and teosinte samples.

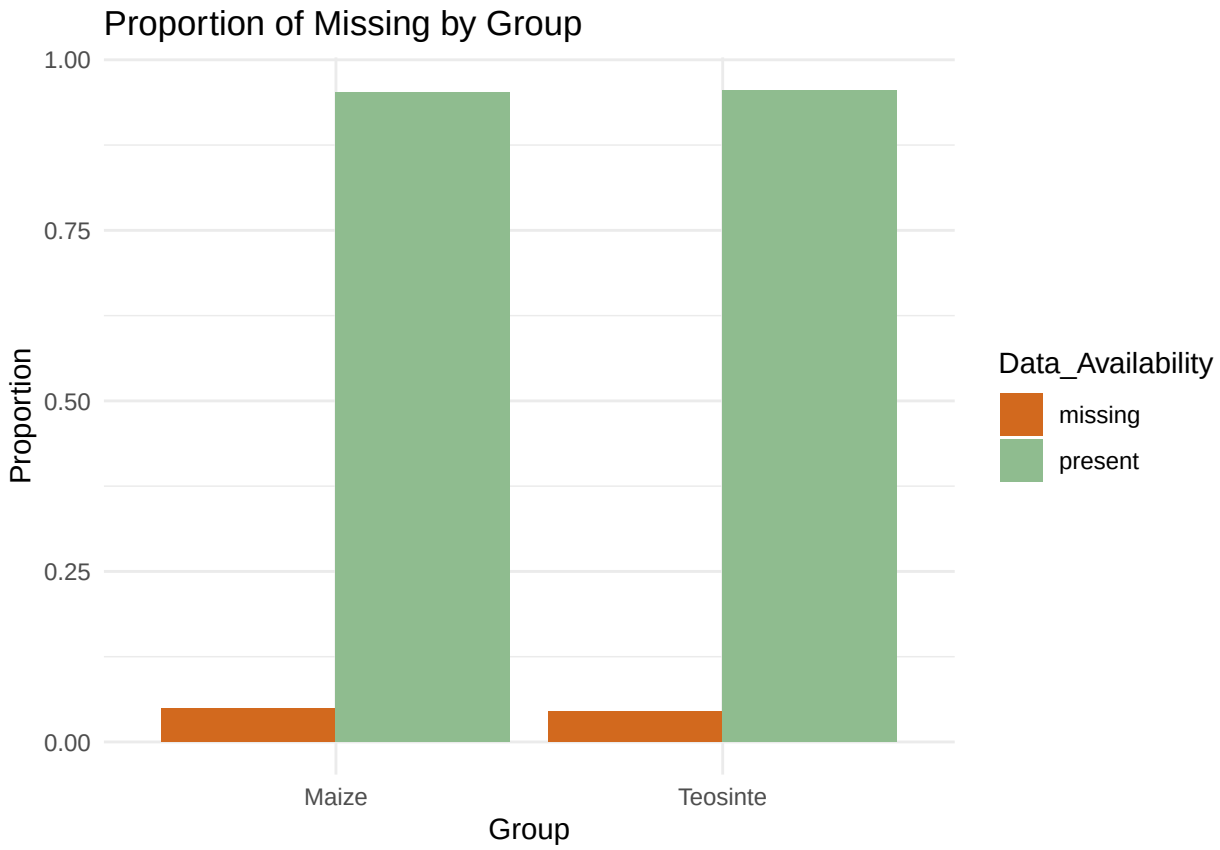
```
# ~~~ missing data ~~~

# # reshape data to long format and filter FOR `?/?"` SNPs
snp_long_missing <- snp_plot2_df |>
  pivot_longer(cols = c(contains("_m"), contains("_t")), names_to = "Sample", values_to = "Genotype") |>
  mutate(
    Chromosome = factor(Chromosome),  # ensure Chromosome is a factor
    Group = ifelse(grepl("_m", Sample), "Maize", "Teosinte"), # assign group
    Data = ifelse(Genotype == "?/?", "missing", "present")
  )
```

```r
# proportion of missing data for each sample
missing_prop_sample <- table(snp_long_missing$Sample, snp_long_missing$Data)
missing_prop_sample <- prop.table(missing_prop_sample, 1)

# Proportion of missing data for each group
missing_prop_group <- table(snp_long_missing$Group, snp_long_missing$Data)
missing_prop_group <- prop.table(missing_prop_group, 1)

print("Proportion of Missing Data for each Sample:")
print(missing_prop_sample)

print("Proportion of Missing Data for each Group:")
print(missing_prop_group)

# convert proportion tables to data frames for plotting
miss_prop_sample_df <- as.data.frame(as.table(missing_prop_sample))
miss_prop_group_df <- as.data.frame(as.table(missing_prop_group))

colnames(miss_prop_sample_df) <- c("Sample", "Data_Availability", "Proportion")
colnames(miss_prop_group_df) <- c("Group", "Data_Availability", "Proportion")
```

```r
# plot for each Sample
plot2c <- ggplot(miss_prop_sample_df, aes(x = Sample, y = Proportion, fill = Data_Availability)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Proportion of Missing Data by Sample",
    x = "Sample (one line = one sample)",
    y = "Proportion") +
  scale_fill_manual(values = c("present" = "darkseagreen", "missing" = "chocolate")) +
  theme_minimal() +
  theme(axis.text.x = element_blank(),
        axis.ticks.x = element_blank())

# plot for each Group
plot2d <- ggplot(miss_prop_group_df, aes(x = Group, y = Proportion, fill = Data_Availability)) +
  geom_bar(stat = "identity", position = "dodge") +
  labs(
    title = "Proportion of Missing by Group",
    x = "Group",
    y = "Proportion") +
  scale_fill_manual(values = c("present" = "darkseagreen", "missing" = "chocolate")) +
  theme_minimal()

plot2c
```


Proportion of Missing Data by Sample
1.00
0.75
0.50
0.25
0.00
Proportion
Sample (one line = one sample)
Data_Availability
missing
present

```
plot2d
```




```
ggsave("Proportion of Missing Data by Sample.png", plot = plot2c)
```

```
## Saving 6.5 x 4.5 in image
```

```
ggsave("Proportion of Missing Data by Group.png", plot = plot2d)
```

```
## Saving 6.5 x 4.5 in image
```

Explanation

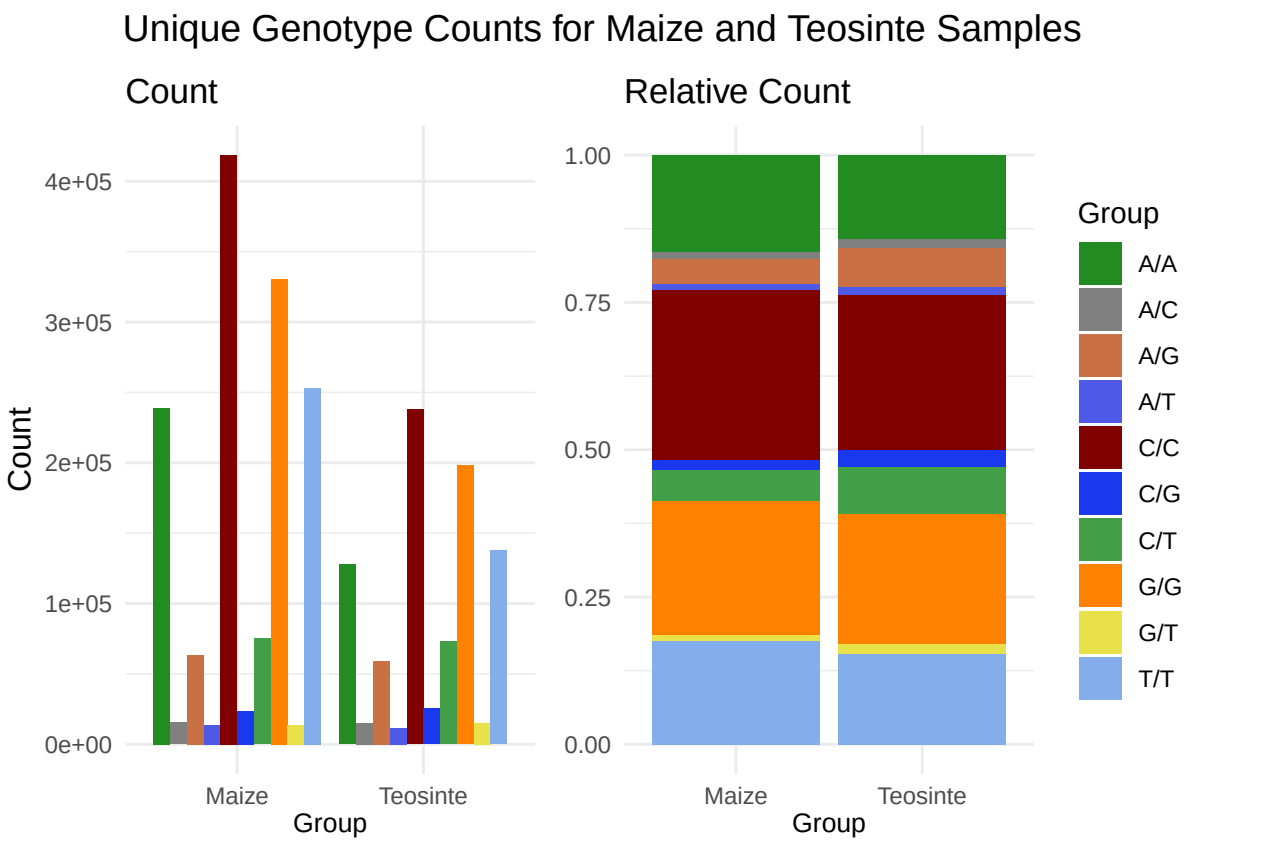
- The same data prep and analysis performed for plot2a and plot2b were done to examine missing genotypes from the samples.
- Plot 2c displays the proportion of missing and present data sites for each sample (one line = one sample); however this figure is hard to read due to the high number of samples.
- Plot 2d groups the data by species (maize & teosinte) and displays the proportion of missing and present data sites in a much simpler way. In this plot, it is obvious that there are significantly more present data than missing in both the maize and teosinte samples, allowing for subsequent analyses to be completed with confidence.

Unique visualization:

```
snp_long_chr_only <- snp_long |>
  filter(Chromosome != c("unknown", "multiple"))

genotype_counts <- snp_long_chr_only |>
  pivot_longer(cols = c(Genotype), names_to = "Genotypes", values_to = "Unique_genotypes") |>
  count(Group, Unique_genotypes)
```

```
# plot results: Stacked bar chart of genotype frequencies
count_plot <- ggplot(genotype_counts, aes(x = Group, y = n, fill = Unique_genotypes)) +
```

```
  geom_bar(stat = "identity", position = "dodge") +
  labs(title = "Genotype Counts by Position (1st vs. 2nd) in Maize & Teosinte",
       x = "Group",
       y = "Count") +
  theme_minimal() +
  theme(legend.position = "none") +
  scale_fill_manual(name = "Group",
                    values = c("forestgreen", "#808080", "#c97044", "#4e5ae7", "#810000", "#1a38ed", "#4

relative_plot <- ggplot(genotype_counts, aes(x = Group, y = n, fill = Unique_genotypes)) +
  geom_bar(stat = "identity", position = "fill") +
  labs(title = "Genotype Counts by Position (1st vs. 2nd) in Maize & Teosinte",
       x = "Group",
       y = "Count") +
  theme_minimal() +
  scale_fill_manual(name = "Group",
                    values = c("forestgreen", "#808080", "#c97044", "#4e5ae7", "#810000", "#1a38ed", "#4


plot3 <- (count_plot +
            labs(title = "Count") +
            theme(
              axis.title.x = element_blank(),
              axis.title.y = element_text(size = 12))) +
  (relative_plot +
     labs(title = "Relative Count") +
     theme(
       axis.title.x = element_blank(),
       axis.title.y = element_blank()
     )) +
  plot_annotation(
    title = "Unique Genotype Counts for Maize and Teosinte Samples",
    theme = theme(
      plot.title = element_text(hjust = 0.5, size = 14)
    )
  ) &
  theme(
    axis.title.x = element_text(size = 10)
  )

plot3
```




```
ggsave("Unique Genotype Counts for Maize and Teosinte Samples.png", plot = plot3)
```

```
## Saving 6.5 x 4.5 in image
```

Explanation

- This figure compares the distribution of unique genotypes between maize and teosinte samples using two different visualizations: Absolute Count (left) vs. Relative Count (right) of nucleotide pairings
- Left Plot: Absolute Count
  - This plot shows the absolute count of different genotypes (e.g., A/A, C/G, G/T, etc.) within each group (Maize vs. Teosinte).
  - Some genotypes, such as C/C and G/G, appear to be much more frequent than others.
  - The maize group has higher absolute counts of some genotypes, suggesting a greater total number of SNPs or samples in this group.
  - There are visible differences in genotype distributions between maize and teosinte, suggesting some genotypes are more common in one group.
- Right Plot: Relative Count
  - This plot represents the relative frequency (proportion) of each genotype within maize and teosinte.
  - Since the total height of the bars is normalized to 1, this allows for easy comparison of genotype composition between groups.
  - While the absolute counts differ (left plot), the relative proportions of genotypes appear similar between maize and teosinte.