Corrected Readme

Manuela Chaves

2024-03-21

During this R assignment, two raw genomic data files ("fang _et_al_genotypes.txt" and "snp_position.txt") were subjected to a file inspection process through Rstudio, to describe and understand the kind of data we are dealing with, as well as to pre-process the data for further analysis.

Files:

fang_et_al_genotypes.txt: a published SNP data set including maize, teosinte (i.e., wild maize), and Tripsacum (a close outgroup to the genus Zea) individuals.

snp_position.txt: an additional data file that includes the SNP id (first column), chromosome location (third column), nucleotide location (fourth column) and other information for the SNPs genotyped in the fang_et_al_genotypes.txt file.

Note:

1. This Readme file implemented corrections from reviewers.Go to "Readme.rmd" to see the first version of the project.
2. The created_files folder contain all created files in this R project in case you need them.

Part 1. Data inspection

Reading .txt files:

- The following command chunk will run the initial files, which are contained in the same repository folder.

```
# Load required libraries
chooseCRANmirror(ind=1)  # Elige un espejo de CRAN
install.packages("ggplot2")
```

```
## Installing package into 'C:/Users/Acer/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'ggplot2' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Acer\AppData\Local\Temp\RtmpSWwHWi\downloaded_packages
```

```
install.packages("tidyverse")
```

```
## Installing package into 'C:/Users/Acer/AppData/Local/R/win-library/4.3'
## (as 'lib' is unspecified)
```

```
## package 'tidyverse' successfully unpacked and MD5 sums checked
##
## The downloaded binary packages are in
##  C:\Users\Acer\AppData\Local\Temp\RtmpSWwHWi\downloaded_packages
```

```
library(readr)
library(data.table)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(tidyr)
```

```
snp <- read_tsv("snp_position.txt", show_col_types = FALSE)
fang <- read_tsv("fang_et_al_genotypes.txt", show_col_types = FALSE)
```

Attributes of the Fang et al & SNP files

- The following chunk shows the commands I used to inspect the data:

```
#Inspecting the Fang file

dim(fang) #dimensions of the file
```

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

```
str(fang[, 1:5]) #structure of the file for the first 5 columns
```

```
## tibble [2,782 x 5] (S3: tbl_df/tbl/data.frame)
##  $ Sample_ID: chr [1:2782] "SL-15" "SL-16" "SL-11" "SL-12" ...
##  $ JG_OTU   : chr [1:2782] "T-aust-1" "T-aust-2" "T-brav-1" "T-brav-2" ...
##  $ Group    : chr [1:2782] "TRIPS" "TRIPS" "TRIPS" "TRIPS" ...
##  $ abph1.20 : chr [1:2782] "?/?" "?/?" "?/?" "?/?" ...
##  $ abph1.22 : chr [1:2782] "?/?" "?/?" "?/?" "?/?" ...
```

```
#summary(fang[, 1:5]) # statistic summary
names(fang) #shows names of all columns in file
```

```
##   [1] "Sample_ID"      "JG_OTU"         "Group"          "abph1.20"
##   [5] "abph1.22"       "ae1.3"          "ae1.4"          "ae1.5"
##   [9] "an1.4"          "ba1.6"          "ba1.9"          "bt2.5"
##  [13] "bt2.7"          "bt2.8"          "Fea2.1"         "Fea2.5"
##  [17] "id1.3"          "lg2.11"         "lg2.2"          "pbf1.1"
##  [21] "pbf1.2"         "pbf1.3"         "pbf1.5"         "pbf1.6"
##  [25] "pbf1.7"         "pbf1.8"         "PZA00003.11"    "PZA00004.2"
##  [29] "PZA00005.8"     "PZA00005.9"     "PZA00006.13"    "PZA00006.14"
##  [33] "PZA00008.1"     "PZA00010.5"     "PZA00013.10"    "PZA00013.11"
##  [37] "PZA00013.9"     "PZA00015.4"     "PZA00017.1"     "PZA00018.5"
##  [41] "PZA00029.11"    "PZA00029.12"    "PZA00030.11"    "PZA00031.5"
##  [45] "PZA00041.3"     "PZA00042.2"     "PZA00042.5"     "PZA00043.7"
##  [49] "PZA00045.1"     "PZA00047.2"     "PZA00049.12"    "PZA00050.9"
##  [53] "PZA00051.2"     "PZA00058.5"     "PZA00058.6"     "PZA00060.2"
##  [57] "PZA00061.1"     "PZA00065.2"     "PZA00069.4"     "PZA00070.5"
##  [61] "PZA00078.2"     "PZA00079.1"     "PZA00081.17"    "PZA00084.2"
##  [65] "PZA00084.3"     "PZA00086.8"     "PZA00088.3"     "PZA00090.2"
##  [69] "PZA00092.1"     "PZA00092.5"     "PZA00093.2"     "PZA00096.26"
##  [73] "PZA00097.13"    "PZA00098.14"    "PZA00100.10"    "PZA00100.12"
##  [77] "PZA00100.14"    "PZA00100.9"     "PZA00103.20"    "PZA00106.9"
##  [81] "PZA00107.18"    "PZA00108.12"    "PZA00108.14"    "PZA00108.15"
##  [85] "PZA00109.3"     "PZA00109.5"     "PZA00111.2"     "PZA00111.4"
##  [89] "PZA00111.5"     "PZA00111.6"     "PZA00111.8"     "PZA00114.3"
##  [93] "PZA00116.2"     "PZA00119.4"     "PZA00120.4"     "PZA00123.1"
##  [97] "PZA00125.2"     "PZA00131.14"    "PZA00132.17"    "PZA00132.18"
## [101] "PZA00132.3"     "PZA00135.6"     "PZA00137.2"     "PZA00139.14"
## [105] "PZA00140.10"    "PZA00140.6"     "PZA00140.9"     "PZA00142.6"
## [109] "PZA00148.2"     "PZA00153.3"     "PZA00153.6"     "PZA00163.4"
## [113] "PZA00164.1"     "PZA00164.2"     "PZA00164.3"     "PZA00166.1"
## [117] "PZA00166.3"     "PZA00170.1"     "PZA00170.3"     "PZA00170.4"
## [121] "PZA00174.1"     "PZA00174.2"     "PZA00175.2"     "PZA00176.8"
## [125] "PZA00177.4"     "PZA00178.3"     "PZA00182.3"     "PZA00182.4"
## [129] "PZA00184.1"     "PZA00184.4"     "PZA00188.1"     "PZA00188.3"
## [133] "PZA00191.5"     "PZA00192.6"     "PZA00192.7"     "PZA00193.2"
## [137] "PZA00198.39"    "PZA00200.11"    "PZA00200.17"    "PZA00200.9"
## [141] "PZA00201.2"     "PZA00204.1"     "PZA00210.1"     "PZA00210.6"
## [145] "PZA00211.7"     "PZA00212.1"     "PZA00213.19"    "PZA00214.1"
## [149] "PZA00216.9"     "PZA00218.1"     "PZA00218.6"     "PZA00219.7"
## [153] "PZA00220.11"    "PZA00220.12"    "PZA00221.7"     "PZA00225.8"
## [157] "PZA00226.7"     "PZA00227.8"     "PZA00230.5"     "PZA00232.24"
## [161] "PZA00234.21"    "PZA00235.6"     "PZA00235.8"     "PZA00237.2"
## [165] "PZA00237.7"     "PZA00237.8"     "PZA00238.3"     "PZA00240.9"
## [169] "PZA00241.6"     "PZA00243.27"    "PZA00245.14"    "PZA00245.16"
## [173] "PZA00245.17"    "PZA00245.18"    "PZA00245.19"    "PZA00249.2"
```

```
## [177] "PZA00250.1"     "PZA00251.1"     "PZA00254.3"     "PZA00255.15"
## [181] "PZA00255.17"    "PZA00256.16"    "PZA00256.21"    "PZA00256.23"
## [185] "PZA00257.11"    "PZA00257.22"    "PZA00261.6"     "PZA00263.14"
## [189] "PZA00266.5"     "PZA00270.3"     "PZA00273.1"     "PZA00274.7"
## [193] "PZA00277.17"    "PZA00277.9"     "PZA00280.14"    "PZA00287.1"
## [197] "PZA00289.11"    "PZA00294.20"    "PZA00296.6"     "PZA00297.2"
## [201] "PZA00297.3"     "PZA00297.4"     "PZA00298.4"     "PZA00298.5"
## [205] "PZA00299.2"     "PZA00300.12"    "PZA00300.13"    "PZA00300.14"
## [209] "PZA00300.16"    "PZA00301.3"     "PZA00303.19"    "PZA00303.21"
## [213] "PZA00307.12"    "PZA00307.14"    "PZA00307.17"    "PZA00309.2"
## [217] "PZA00310.5"     "PZA00314.6"     "PZA00314.8"     "PZA00315.1"
## [221] "PZA00315.6"     "PZA00318.2"     "PZA00323.3"     "PZA00323.4"
## [225] "PZA00326.16"    "PZA00326.18"    "PZA00326.19"    "PZA00332.8"
## [229] "PZA00332.9"     "PZA00334.2"     "PZA00335.12"    "PZA00337.3"
## [233] "PZA00337.4"     "PZA00337.5"     "PZA00342.9"     "PZA00344.10"
## [237] "PZA00345.15"    "PZA00346.1"     "PZA00346.2"     "PZA00346.3"
## [241] "PZA00349.3"     "PZA00349.5"     "PZA00350.2"     "PZA00352.22"
## [245] "PZA00355.1"     "PZA00355.2"     "PZA00356.9"     "PZA00364.5"
## [249] "PZA00364.6"     "PZA00367.2"     "PZA00369.1"     "PZA00370.1"
## [253] "PZA00370.5"     "PZA00380.5"     "PZA00380.7"     "PZA00381.3"
## [257] "PZA00381.4"     "PZA00381.5"     "PZA00382.17"    "PZA00385.3"
## [261] "PZA00386.3"     "PZA00390.6"     "PZA00391.2"     "PZA00392.3"
## [265] "PZA00392.4"     "PZA00393.1"     "PZA00393.4"     "PZA00394.11"
## [269] "PZA00395.1"     "PZA00395.2"     "PZA00396.12"    "PZA00401.11"
## [273] "PZA00401.6"     "PZA00406.1"     "PZA00407.9"     "PZA00408.7"
## [277] "PZA00409.3"     "PZA00411.1"     "PZA00411.4"     "PZA00411.5"
## [281] "PZA00413.17"    "PZA00413.18"    "PZA00413.21"    "PZA00417.2"
## [285] "PZA00417.3"     "PZA00419.1"     "PZA00420.4"     "PZA00422.2"
## [289] "PZA00422.5"     "PZA00422.6"     "PZA00423.16"    "PZA00423.17"
## [293] "PZA00424.1"     "PZA00425.4"     "PZA00425.9"     "PZA00429.1"
## [297] "PZA00433.5"     "PZA00436.7"     "PZA00439.6"     "PZA00440.1"
## [301] "PZA00442.3"     "PZA00442.4"     "PZA00442.5"     "PZA00442.6"
## [305] "PZA00444.1"     "PZA00444.5"     "PZA00445.18"    "PZA00449.2"
## [309] "PZA00452.4"     "PZA00458.6"     "PZA00459.5"     "PZA00460.3"
## [313] "PZA00460.5"     "PZA00460.7"     "PZA00462.2"     "PZA00463.3"
## [317] "PZA00466.1"     "PZA00468.11"    "PZA00468.7"     "PZA00470.1"
## [321] "PZA00471.2"     "PZA00471.3"     "PZA00471.4"     "PZA00472.2"
## [325] "PZA00477.10"    "PZA00477.11"    "PZA00477.5"     "PZA00477.9"
## [329] "PZA00478.10"    "PZA00478.11"    "PZA00478.7"     "PZA00478.9"
## [333] "PZA00480.10"    "PZA00481.7"     "PZA00484.5"     "PZA00485.2"
## [337] "PZA00486.2"     "PZA00487.16"    "PZA00487.24"    "PZA00487.26"
## [341] "PZA00489.1"     "PZA00493.1"     "PZA00493.2"     "PZA00493.5"
## [345] "PZA00495.3"     "PZA00495.4"     "PZA00495.6"     "PZA00496.1"
## [349] "PZA00497.1"     "PZA00497.4"     "PZA00498.4"     "PZA00499.10"
## [353] "PZA00499.12"    "PZA00499.3"     "PZA00501.12"    "PZA00501.14"
## [357] "PZA00502.5"     "PZA00503.5"     "PZA00504.1"     "PZA00504.2"
## [361] "PZA00505.4"     "PZA00505.8"     "PZA00510.2"     "PZA00510.3"
## [365] "PZA00514.1"     "PZA00514.6"     "PZA00514.7"     "PZA00515.14"
## [369] "PZA00516.3"     "PZA00517.6"     "PZA00522.12"    "PZA00523.2"
## [373] "PZA00525.16"    "PZA00525.2"     "PZA00527.6"     "PZA00527.9"
## [377] "PZA00529.3"     "PZA00531.1"     "PZA00533.3"     "PZA00533.4"
## [381] "PZA00533.5"     "PZA00533.6"     "PZA00534.2"     "PZA00536.2"
## [385] "PZA00538.12"    "PZA00538.16"    "PZA00538.8"     "PZA00543.2"
## [389] "PZA00543.4"     "PZA00543.5"     "PZA00545.21"    "PZA00545.22"
```

```
## [393] "PZA00545.4"     "PZA00547.13"    "PZA00547.18"    "PZA00552.4"
## [397] "PZA00560.1"     "PZA00560.2"     "PZA00562.4"     "PZA00565.3"
## [401] "PZA00566.5"     "PZA00568.19"    "PZA00573.3"     "PZA00578.1"
## [405] "PZA00579.6"     "PZA00582.4"     "PZA00586.1"     "PZA00587.3"
## [409] "PZA00587.6"     "PZA00588.2"     "PZA00588.4"     "PZA00589.10"
## [413] "PZA00589.8"     "PZA00589.9"     "PZA00593.2"     "PZA00595.3"
## [417] "PZA00600.11"    "PZA00603.1"     "PZA00608.1"     "PZA00608.5"
## [421] "PZA00610.18"    "PZA00610.9"     "PZA00613.22"    "PZA00614.12"
## [425] "PZA00615.3"     "PZA00615.6"     "PZA00615.8"     "PZA00617.16"
## [429] "PZA00618.22"    "PZA00620.2"     "PZA00621.2"     "PZA00622.1"
## [433] "PZA00622.2"     "PZA00623.2"     "PZA00626.3"     "PZA00626.4"
## [437] "PZA00630.9"     "PZA00636.5"     "PZA00636.6"     "PZA00637.4"
## [441] "PZA00639.12"    "PZA00639.13"    "PZA00639.15"    "PZA00641.7"
## [445] "PZA00641.8"     "PZA00644.11"    "PZA00647.9"     "PZA00650.8"
## [449] "PZA00654.10"    "PZA00654.12"    "PZA00655.1"     "PZA00656.15"
## [453] "PZA00656.16"    "PZA00656.18"    "PZA00656.4"     "PZA00658.19"
## [457] "PZA00658.23"    "PZA00662.3"     "PZA00665.6"     "PZA00667.1"
## [461] "PZA00672.6"     "PZA00672.8"     "PZA00673.2"     "PZA00674.3"
## [465] "PZA00676.2"     "PZA00680.1"     "PZA00680.3"     "PZA00682.2"
## [469] "PZA00684.12"    "PZA00686.8"     "PZA00692.5"     "PZA00693.3"
## [473] "PZA00695.1"     "PZA00698.4"     "PZA00700.3"     "PZA00704.11"
## [477] "PZA00705.5"     "PZA00706.16"    "PZA00710.1"     "PZA00710.16"
## [481] "PZA00712.4"     "PZA00715.3"     "PZA00717.14"    "PZA00719.1"
## [485] "PZA00719.2"     "PZA00719.3"     "PZA00720.2"     "PZA00720.3"
## [489] "PZA00721.4"     "PZA00721.5"     "PZA00725.4"     "PZA00726.6"
## [493] "PZA00726.7"     "PZA00726.9"     "PZA00727.11"    "PZA00727.12"
## [497] "PZA00729.18"    "PZA00729.19"    "PZA00730.2"     "PZA00731.6"
## [501] "PZA00731.7"     "PZA01104.1"     "PZA01149.1"     "PZA01149.3"
## [505] "PZA01182.1"     "PZA01240.1"     "PZA01240.2"     "PZA01420.1"
## [509] "PZA01420.2"     "PZA01420.3"     "PZA01474.2"     "PZA01637.2"
## [513] "PZA01637.3"     "PZA01637.4"     "PZA01725.1"     "PZA01725.2"
## [517] "PZA01782.2"     "PZA01782.3"     "PZA01782.4"     "PZA02789.31"
## [521] "PZA02789.36"    "PZA02791.6"     "PZA02792.16"    "PZA02792.9"
## [525] "PZA02806.4"     "PZA02806.9"     "PZA02807.5"     "PZA02808.12"
## [529] "PZA02808.16"    "PZA02819.35"    "PZA02820.6"     "PZA02822.2"
## [533] "PZA02824.1"     "PZA02824.3"     "PZA02825.8"     "PZA02831.5"
## [537] "PZA02837.5"     "PZA02844.1"     "PZA02850.18"    "PZA02850.4"
## [541] "PZA02853.10"    "PZA02853.7"     "PZA02856.1"     "PZA02862.3"
## [545] "PZA02865.11"    "PZA02869.2"     "PZA02869.8"     "PZA02872.1"
## [549] "PZA02872.3"     "PZA02878.12"    "PZA02888.3"     "PZA02890.3"
## [553] "PZA02890.4"     "PZA02890.5"     "PZA02894.1"     "PZA02897.12"
## [557] "PZA02906.12"    "PZA02906.7"     "PZA02921.9"     "PZA02923.7"
## [561] "PZA02927.1"     "PZA02938.5"     "PZA02939.6"     "PZA02940.3"
## [565] "PZA02941.3"     "PZA02941.6"     "PZA02941.8"     "PZA02947.2"
## [569] "PZA02948.19"    "PZA02948.21"    "PZA02948.22"    "PZA02949.22"
## [573] "PZA02949.26"    "PZA02952.10"    "PZA02954.2"     "PZA02955.3"
## [577] "PZA02958.17"    "PZA02959.7"     "PZA02961.1"     "PZA02962.13"
## [581] "PZA02963.5"     "PZA02966.11"    "PZA02968.4"     "PZA02969.11"
## [585] "PZA02970.9"     "PZA02972.1"     "PZA02982.5"     "PZA02982.6"
## [589] "PZA02983.38"    "PZA02984.7"     "PZA02988.2"     "PZA02993.5"
## [593] "PZA02997.16"    "PZA02997.19"    "PZA03001.15"    "PZA03001.18"
## [597] "PZA03001.9"     "PZA03009.5"     "PZA03009.6"     "PZA03009.7"
## [601] "PZA03009.8"     "PZA03011.6"     "PZA03012.10"    "PZA03013.7"
## [605] "PZA03013.8"     "PZA03014.10"    "PZA03014.21"    "PZA03014.24"
```

```
## [609] "PZA03017.10"   "PZA03017.11"   "PZA03024.16"   "PZA03024.18"
## [613] "PZA03024.7"    "PZA03028.5"    "PZA03032.16"   "PZA03034.1"
## [617] "PZA03035.5"    "PZA03037.8"    "PZA03037.9"    "PZA03041.8"
## [621] "PZA03042.1"    "PZA03042.5"    "PZA03046.2"    "PZA03046.3"
## [625] "PZA03047.12"   "PZA03047.20"   "PZA03047.22"   "PZA03048.16"
## [629] "PZA03048.17"   "PZA03049.23"   "PZA03051.1"    "PZA03051.3"
## [633] "PZA03052.15"   "PZA03054.3"    "PZA03054.5"    "PZA03058.17"
## [637] "PZA03062.15"   "PZA03062.7"    "PZA03063.17"   "PZA03063.18"
## [641] "PZA03064.6"    "PZA03067.17"   "PZA03067.20"   "PZA03068.11"
## [645] "PZA03068.13"   "PZA03069.6"    "PZA03073.23"   "PZA03073.24"
## [649] "PZA03074.24"   "PZA03078.33"   "PZA03081.1"    "PZA03081.10"
## [653] "PZA03081.11"   "PZA03081.13"   "PZA03081.6"    "PZA03083.7"
## [657] "PZA03089.12"   "PZA03090.31"   "PZA03092.7"    "PZA03094.18"
## [661] "PZA03094.6"    "PZA03095.1"    "PZA03095.2"    "PZA03095.3"
## [665] "PZA03097.4"    "PZA03097.7"    "PZA03097.9"    "PZA03102.10"
## [669] "PZA03102.2"    "PZA03102.9"    "PZA03137.1"    "PZA03172.2"
## [673] "PZA03223.3"    "PZA03258.2"    "PZA03283.2"    "PZA03284.3"
## [677] "PZA03290.1"    "PZA03290.2"    "PZA03295.4"    "PZA03296.6"
## [681] "PZA03296.7"    "PZA03298.1"    "PZA03298.2"    "PZA03301.2"
## [685] "PZA03301.4"    "PZA03302.1"    "PZA03305.6"    "PZA03305.7"
## [689] "PZA03311.2"    "PZA03311.3"    "PZA03311.4"    "PZA03311.5"
## [693] "PZA03312.1"    "PZA03312.2"    "PZA03316.2"    "PZA03317.1"
## [697] "PZA03319.3"    "PZA03319.4"    "PZA03320.3"    "PZA03320.4"
## [701] "PZA03328.5"    "PZA03329.1"    "PZA03329.2"    "PZA03333.3"
## [705] "PZA03335.2"    "PZA03335.3"    "PZA03337.1"    "PZA03338.5"
## [709] "PZA03340.2"    "PZA03342.2"    "PZA03344.4"    "PZA03344.5"
## [713] "PZA03344.6"    "PZA03345.1"    "PZA03345.2"    "PZA03345.4"
## [717] "PZA03347.1"    "PZA03348.1"    "PZA03349.1"    "PZA03349.9"
## [721] "PZA03767.1"    "PZA03767.4"    "PZA03767.5"    "PZA03773.2"
## [725] "PZA03773.3"    "PZA03774.1"    "PZA03774.10"   "PZA03774.2"
## [729] "PZA03774.4"    "PZA03774.5"    "PZA03774.6"    "PZA03774.8"
## [733] "PZA03774.9"    "PZA03775.1"    "PZA03775.11"   "PZA03775.2"
## [737] "PZA03775.3"    "PZA03775.4"    "PZA03775.6"    "PZA03775.7"
## [741] "PZA03775.8"    "PZA03775.9"    "PZA03781.1"    "PZA03781.2"
## [745] "PZA03781.3"    "PZA03781.4"    "PZA03781.5"    "PZA03781.6"
## [749] "PZA03781.7"    "PZA03781.8"    "PZA03782.1"    "PZA03782.3"
## [753] "PZA03786.1"    "PZA03786.2"    "PZA03789.1"    "PZA03789.2"
## [757] "PZA03789.4"    "PZB00011.4"    "PZB00011.5"    "PZB00041.2"
## [761] "PZB00041.4"    "PZB00049.2"    "PZB00049.4"    "PZB00049.7"
## [765] "PZB00055.1"    "PZB00060.4"    "PZB00062.6"    "PZB00062.7"
## [769] "PZB00062.8"    "PZB00067.2"    "PZB00067.3"    "PZB00067.4"
## [773] "PZB00067.5"    "PZB00078.1"    "PZB00081.2"    "PZB00081.4"
## [777] "PZB00081.5"    "PZB00081.7"    "PZB00092.1"    "PZB00092.4"
## [781] "PZB00093.3"    "PZB00093.4"    "PZB00093.6"    "PZB00096.2"
## [785] "PZB00096.3"    "PZB00136.3"    "PZB00140.1"    "PZB00145.2"
## [789] "PZB00149.2"    "PZB00149.4"    "PZB00153.1"    "PZB00153.2"
## [793] "PZB00153.3"    "PZB00153.5"    "PZB00160.1"    "PZB00160.2"
## [797] "PZB00160.4"    "PZB00165.2"    "PZB00165.6"    "PZB00169.4"
## [801] "PZB00169.6"    "PZB00175.1"    "PZB00175.2"    "PZB00175.3"
## [805] "PZB00175.4"    "PZB00175.5"    "PZB00180.1"    "PZB00180.2"
## [809] "PZB00183.3"    "PZB00188.6"    "PZB00207.3"    "PZB00221.3"
## [813] "PZB00221.8"    "PZB00229.3"    "PZB00232.1"    "PZB00232.2"
## [817] "PZB00232.4"    "PZB00232.5"    "PZB00379.3"    "PZB00379.4"
## [821] "PZB00379.5"    "PZB00393.7"    "PZB00409.3"    "PZB00416.2"
```

```
## [825] "PZB00416.5"     "PZB00454.2"     "PZB00454.3"     "PZB00454.4"
## [829] "PZB00454.5"     "PZB00498.2"     "PZB00498.4"     "PZB00598.1"
## [833] "PZB00598.2"     "PZB00603.3"     "PZB00603.4"     "PZB00603.5"
## [837] "PZB00607.2"     "PZB00761.1"     "PZB00761.2"     "PZB00849.2"
## [841] "PZB00849.3"     "PZB00849.4"     "PZB00859.1"     "PZB01109.2"
## [845] "PZB01109.3"     "PZB01110.1"     "PZB01110.2"     "PZB01110.3"
## [849] "PZB01111.6"     "PZB01111.7"     "PZB01111.8"     "PZB01112.3"
## [853] "PZB01112.4"     "PZB01112.5"     "PZB01112.6"     "PZB01113.4"
## [857] "PZB01114.1"     "PZB01114.3"     "PZB01115.1"     "PZB01115.5"
## [861] "PZB01115.6"     "PZB01116.2"     "PZB01221.1"     "PZB01222.1"
## [865] "PZB01222.3"     "PZB01223.3"     "PZB01223.4"     "PZB01223.7"
## [869] "PZB01225.1"     "PZB01225.2"     "PZB01225.4"     "PZB01228.1"
## [873] "PZB01228.3"     "PZB01228.4"     "PZB01233.2"     "PZB01233.3"
## [877] "PZB01238.5"     "PZB01238.6"     "PZB01427.1"     "PZB01427.3"
## [881] "PZB01463.2"     "PZB01463.3"     "PZB01463.4"     "PZD00003.1"
## [885] "PZD00003.3"     "PZD00007.1"     "PZD00008.3"     "PZD00011.1"
## [889] "PZD00011.3"     "PZD00011.4"     "PZD00012.1"     "PZD00012.2"
## [893] "PZD00012.3"     "PZD00012.4"     "PZD00012.5"     "PZD00013.3"
## [897] "PZD00013.4"     "PZD00014.3"     "PZD00017.1"     "PZD00019.1"
## [901] "PZD00020.2"     "PZD00020.3"     "PZD00020.4"     "PZD00020.6"
## [905] "PZD00021.2"     "PZD00021.4"     "PZD00021.5"     "PZD00022.1"
## [909] "PZD00022.3"     "PZD00022.4"     "PZD00024.2"     "PZD00025.1"
## [913] "PZD00025.2"     "PZD00030.1"     "PZD00030.4"     "PZD00030.5"
## [917] "PZD00030.6"     "PZD00034.3"     "PZD00043.1"     "PZD00043.2"
## [921] "PZD00043.3"     "PZD00043.4"     "PZD00044.2"     "PZD00044.3"
## [925] "PZD00044.4"     "PZD00045.1"     "PZD00045.2"     "PZD00045.3"
## [929] "PZD00045.4"     "PZD00049.3"     "PZD00049.4"     "PZD00049.5"
## [933] "PZD00051.1"     "PZD00052.3"     "PZD00052.4"     "PZD00062.2"
## [937] "PZD00066.1"     "PZD00067.1"     "PZD00067.2"     "PZD00067.3"
## [941] "PZD00068.1"     "PZD00069.2"     "PZD00069.3"     "PZD00069.4"
## [945] "PZD00069.5"     "PZD00073.1"     "PZD00073.2"     "PZD00073.6"
## [949] "PZD00074.1"     "PZD00075.1"     "PZD00075.2"     "PZD00076.1"
## [953] "PZD00076.2"     "PZD00076.4"     "PZD00077.10"    "PZD00077.5"
## [957] "PZD00077.7"     "PZD00077.8"     "PZD00078.2"     "Ra2_ORF.1"
## [961] "Ra2_ORF.2"      "Ra2_ORF.4"      "Ra2_promoter.1" "Ra2_promoter.2"
## [965] "Ra2_promoter.3" "sh2.5"          "sh2.6"          "sh2.7"
## [969] "sh2.9"          "su1.4"          "su1.5"          "su1.7"
## [973] "tb1.17"         "tb1.18"         "tb1.19"         "tb1.5"
## [977] "te1.3"          "te1.4"          "zagl1.1"        "zagl1.6"
## [981] "zap1.2"         "zen1.1"         "zen1.2"         "zen1.4"
## [985] "zfl2.6"         "zmm3.4"
```

```
head(fang) #shows first lines of file
```

```
## # 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(fang) #shows last lines of file
```

```
## # 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>, ...
```

```
na_fang <- grepl("?/?", fang, ignore.case = TRUE) #check for missing values

#Inspecting the SNP file
dim(snp)
```

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

```
str(snp)
```

```
## spc_tbl_ [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()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

```
summary(snp)
```

```
##     SNP_ID          cdv_marker_id    Chromosome          Position
##  Length:983         Min.   : 3463   Length:983         Length:983
##  Class :character   1st Qu.: 3978   Class :character   Class :character
##  Mode  :character   Median : 5723   Mode  :character   Mode  :character
##                     Mean   : 5925
##                     3rd Qu.: 6629
##                     Max.   :12480
##    alt_pos          mult_positions       amplicon         cdv_map_feature.name
##  Length:983         Length:983         Length:983         Length:983
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##      gene           candidate/random   Genaissance_daa_id Sequenom_daa_id
##  Length:983         Length:983         Min.   : 7649      Min.   :10474
##  Class :character   Class :character   1st Qu.: 7906      1st Qu.:10784
##  Mode  :character   Mode  :character   Median : 8173      Median :11110
##                                        Mean   : 8524      Mean   :11122
##                                        3rd Qu.: 9834      3rd Qu.:11420
##                                        Max.   :10104      Max.   :11829
##  count_amplicons    count_cmf        count_gene
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000
##  Median :1.0000   Median :1.0000   Median :1.0000
##  Mean   :0.5768   Mean   :0.5483   Mean   :0.5565
##  3rd Qu.:1.0000   3rd Qu.:1.0000   3rd Qu.:1.0000
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000
```

```
names(snp)
```

```
##  [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"
```

```
head(snp)
```

```
## # 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>
```

```
tail(snp)
```

```
## # 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>
```

```
na_snp <- grepl("NA", snp, ignore.case = TRUE)
```

Assessing inspection

1. Fang:

- Number of rows = 2782 and columns = 986
- Data structure shows character type (col_character).
- The summary command is not very useful for the Fang file since it recognizes the data as characters. That is why I commented out this line.
- Tail and head will display part of the data nicely organized in a printed data frame (tibble)
- We can check the names of all the columns in the file with "names()"
- To check for missing values in the Fang file we could use $ colSums(is.na(fang)), which will return a logical matrix of the same dimensions of the file, and sums the TRUE values in each column, TRUE meaning NA. This command returns all 0, meaning all FALSE (not NA). However, we know from the previous UNIX assignment, that the missing data in Fang is delimited as "?/?", in which case

we can use na_fang <- grepl("?/?", fang, ignore.case = TRUE), which will create a data frame with insensitive search, meaning any combination of the missing value. This time we get that there is at least 1 missing data in every column (all TRUE values).

2. SNP:

- Number of rows = 983 and columns = 15
- Data structure includes characters (col_character) and numeric (col_double) types.
- Summary shows statistics for numeric columns.
- Tail and head will display part of the data nicely organized in a printed data frame (tibble)
- We can check the names of all the columns in the file with names()
- Missing values in the SNP file are delimited by NA, in this case we can use the colSums(is.na(snp)) command, and it will return quickly the count of missing data and the column. However this command is sensitive, meaning that if there is any NA value in different notation, the command will not count it. Thus, we can use the na_snp <- grepl("NA", snp, ignore.case = TRUE) to be sure of the missing data results.

Part 2. Data processing

Extracting data

- We need to get columns 1, 2, and 3 for SNP_ID, Chromosome and Position respectively.

```
# Extract the indices of columns matching the pattern, and order by SNP ID.
extracted_columns <- snp[, c(1, 3, 4)]
extracted_columns <- arrange(extracted_columns, SNP_ID)
```

- Extract specifically the Maize (ZMM) and Teosinte (ZMP) groups along with the SNP_ID (in columns) of the Fang file.

```
#Filter the rows with ZMM maize groups
zmm <- filter(fang, startsWith(Group, 'ZMM'))

#transpose file, add back column and row names.
t_zmm <- transpose(zmm)
rownames(t_zmm) <- colnames(zmm)
colnames(t_zmm) <- rownames(zmm)
setDT(t_zmm, keep.rownames = TRUE)
colnames(t_zmm)[colnames(t_zmm) == 'rn'] <- 'SNP_ID'

#Teosinte
zmp <- filter(fang, startsWith(Group, 'ZMP'))
t_zmp <- transpose(zmp)
rownames(t_zmp) <- colnames(zmp)
colnames(t_zmp) <- rownames(zmp)
setDT(t_zmp, keep.rownames = TRUE)
colnames(t_zmp)[colnames(t_zmp) == 'rn'] <- 'SNP_ID'
```

Merging data files

- Now we need to merge the files snp_columns.txt with the transpose data for Maize and Teosinte.

```
# Merging Maize file
maize_snp <- merge(extracted_columns, t_zmm, by = 'SNP_ID')
# Merging Teosinte file
teosinte_snp <- merge(extracted_columns, t_zmp, by = 'SNP_ID')
```

Right now we count with a merged file organized as we need it according to the assignment instructions: column 1 = SNP-ID column 2 = Chromosome column 3 = Position column 4:978 = genotype

Creating the files for the assignment:

The assignment is requesting us to create the following files:

Maize files

1. 10 files (1 for each chromosome) with SNPs ordered based on increasing position values and with missing data encoded by this symbol: ?

- The data is naturally encoded ? for missing data.
- For these files we need to order the SNP_ID according to Position in increasing order.
- The Position column contains non-numerical values (unknown, multiple and empty) which will interfere when trying to order the data based on increasing/decreasing values. Therefore, I am taking those values out and creating a clean version (num_open_merged_zmm), which will be used to obtain the file that we need. It uses the grepl() function to identify rows where the Position column consists entirely of digits.
- This chunk first converts the Position column of open_merged_zmm dataframe to numeric, then converts it to numeric, later it orders it based on the values of the Position column, and finally prints the ordered numeric values.
- In order to check if the ordering worked, we need to make sure our data is numeric, that is why I used twice the as.numeric command.
- Finally, We can check if the data is organized in increasing Position (column 3) by converting the matrix array into a data frame (if needed) and printing a TRUE result if the data is actually increasing. Result printed TRUE, meaning it is increasing.

```
# Let's extract the non-numerical values from the 'Position' column
num_open_merged_zmm <- maize_snp[grepl("^\\d+$", maize_snp$Position), ]

# Convert 'Position' column into numeric
convnum_open_merged_zmm <- as.numeric(num_open_merged_zmm$Position)

# Order the numeric values by the 'Position' column
inc_zmm <- num_open_merged_zmm[order(convnum_open_merged_zmm), ]

# Let's make sure the position column is numeric
num_inc_zmm <- as.numeric(inc_zmm$Position)

# Check if the 'Position' column is in ascending order
is_ordered <- all(diff(num_inc_zmm) >= 0)
```

```r
# Print the result
print(is_ordered)
```

```
## [1] TRUE
```

- We now need to create 1 file for each chromosome (1:10) for the organized file. We can do that with a for loop command that will store automatically the .txt files for the corresponding chromosome.

```r
# Get unique chromosome numbers
chromosomeszmm <- unique(inc_zmm$Chromosome)

# Loop through each chromosome
for (chr in chromosomeszmm) {
  # Filter data for the current chromosome
  chr_data <- inc_zmm[inc_zmm$Chromosome == chr, ]

  # Write the filtered data to a separate file
  file_name <- paste0("maize_increasing_chr_", chr, ".txt")
  write.table(chr_data, file = file_name, sep = "\t", quote = FALSE, row.names = FALSE)
}
```

2. 10 files (1 for each chromosome) with SNPs ordered based on decreasing position values and with missing data encoded by this symbol: -

- For these files we need to order the SNP_ID according to Position in decreasing order. We can use the numeric clean file that we got in the previous item (convnum_open_merged_zmm).
- Another reason why we convert the non-numeric values to numeric is that we could not use the unary operator - (negation) on the data. But once the data is converted to numeric there is no problem.
- When checking for decreasing order, we get FALSE, meaning the order is not ascending.

```r
# Order the original data frame by the numeric values in 'Position' column in decreasing order
dec_zmm <- num_open_merged_zmm[order(-convnum_open_merged_zmm), ]

# Let's make sure the position column is numeric
num_dec_zmm <- as.numeric(dec_zmm$Position)

# Check if the 'Position' column is in decreasing order
is_ordered_dec <- all(diff(num_dec_zmm) >= 0)

# Print the result
print(is_ordered_dec)
```

```
## [1] FALSE
```

- Now we need to change the encoded symbol ?/? for -/-.We can do it with lapply command. However, this transforms the data into a list and we need to convert back to data frame.
- Then we can create 1 file for each chromosome (1:10) for the organized file as we did previously.

```r
# Using lapply to replace "?" with "-" for the entire dataframe
encoded_dec_zmm <- lapply(dec_zmm, function(x) gsub("\\?", "-", x))

# Convert the list back to a dataframe
encoded_dec_zmm_df <- as.data.frame(encoded_dec_zmm)

# Now proceed with writing data to files for each chromosome
# Get unique chromosome numbers
chromosomeszmm_dec <- unique(encoded_dec_zmm_df$Chromosome)

# Loop through each chromosome
for (chr in chromosomeszmm_dec) {
  chr_data <- encoded_dec_zmm_df[encoded_dec_zmm_df$Chromosome == chr, ]
  file_name <- paste0("maize_decreasing_chr_", chr, ".txt")
  write.table(chr_data, file = file_name, sep = "\t", quote = FALSE, row.names = FALSE)
}
```

Teosinte files

3. 10 files (1 for each chromosome) with SNPs ordered based on increasing position values and with missing data encoded by this symbol: ?

- For these files we just need to replace the commands we used for item 1 with the respective data from Teosinte (zmp)
- When checking for increasing order result is TRUE.
- As a reminder, the data is naturally encoded with "?" for missing values.

```r
# Let's extract the non-numerical values from the 'Position' column
num_open_merged_zmp <- teosinte_snp[grepl("^\\d+$", teosinte_snp$Position), ]

# Convert 'Position' column into numeric
convnum_open_merged_zmp <- as.numeric(num_open_merged_zmp$Position)

# Order the numeric values by the 'Position' column
inc_zmp <- num_open_merged_zmp[order(convnum_open_merged_zmp), ]

# Let's make sure the position column is numeric
num_inc_zmp <- as.numeric(inc_zmp$Position)

# Check if the 'Position' column is in ascending order
is_ordered <- all(diff(num_inc_zmp) >= 0)

# Print the result
print(is_ordered)
```

```
## [1] TRUE
```

```r
# Get unique chromosome numbers
chromosomeszmp <- unique(inc_zmp$Chromosome)

# Loop through each chromosome
for (chr in chromosomeszmp) {
```

```
  # Filter data for the current chromosome
  chr_data <- inc_zmp[inc_zmp$Chromosome == chr, ]

  # Write the filtered data to a separate file
  file_name <- paste0("teosinte_increasing_chr_", chr, ".txt")
  write.table(chr_data, file = file_name, sep = "\t", quote = FALSE, row.names = FALSE)
}
```

4. 10 files (1 for each chromosome) with SNPs ordered based on decreasing position values and with missing data encoded by this symbol: -

- For these files we just need to replace the commands we used for item 2 with the respective data from Teosinte (zmp)
- When checking for increasing order result is FALSE.

```
# Order the original data frame by the numeric values in 'Position' column in decreasing order
dec_zmp <- num_open_merged_zmp[order(-convnum_open_merged_zmp), ]

# Let's make sure the position column is numeric
num_dec_zmp <- as.numeric(dec_zmp$Position)

# Check if the 'Position' column is in decreasing order
is_ordered_dec <- all(diff(num_dec_zmp) >= 0)

# Print the result
print(is_ordered_dec)
```

```
## [1] FALSE
```

```
# Using lapply to replace "?" with "-" for the entire dataframe
encoded_dec_zmp <- lapply(dec_zmp, function(x) gsub("\\?", "-", x))

# Convert the list back to a dataframe
encoded_dec_zmp_df <- as.data.frame(encoded_dec_zmp)

# Now proceed with writing data to files for each chromosome
# Get unique chromosome numbers
chromosomeszmp_dec <- unique(encoded_dec_zmp_df$Chromosome)

# Loop through each chromosome
for (chr in chromosomeszmp_dec) {
  chr_data <- encoded_dec_zmp_df[encoded_dec_zmp_df$Chromosome == chr, ]
  file_name <- paste0("teosinte_decreasing_chr_", chr, ".txt")
  write.table(chr_data, file = file_name, sep = "\t", quote = FALSE, row.names = FALSE)
}

# If you want to check the files you can use
# teosinte_dec_1 = read.table("C:/Users/mchavesm/Box/Rass/R_assignment/teosinte_decreasing_chr_1.txt",
```

Conclusion:

We have successfully process our 40 files.

Part 3. Data visualization

SNPs per chromosome

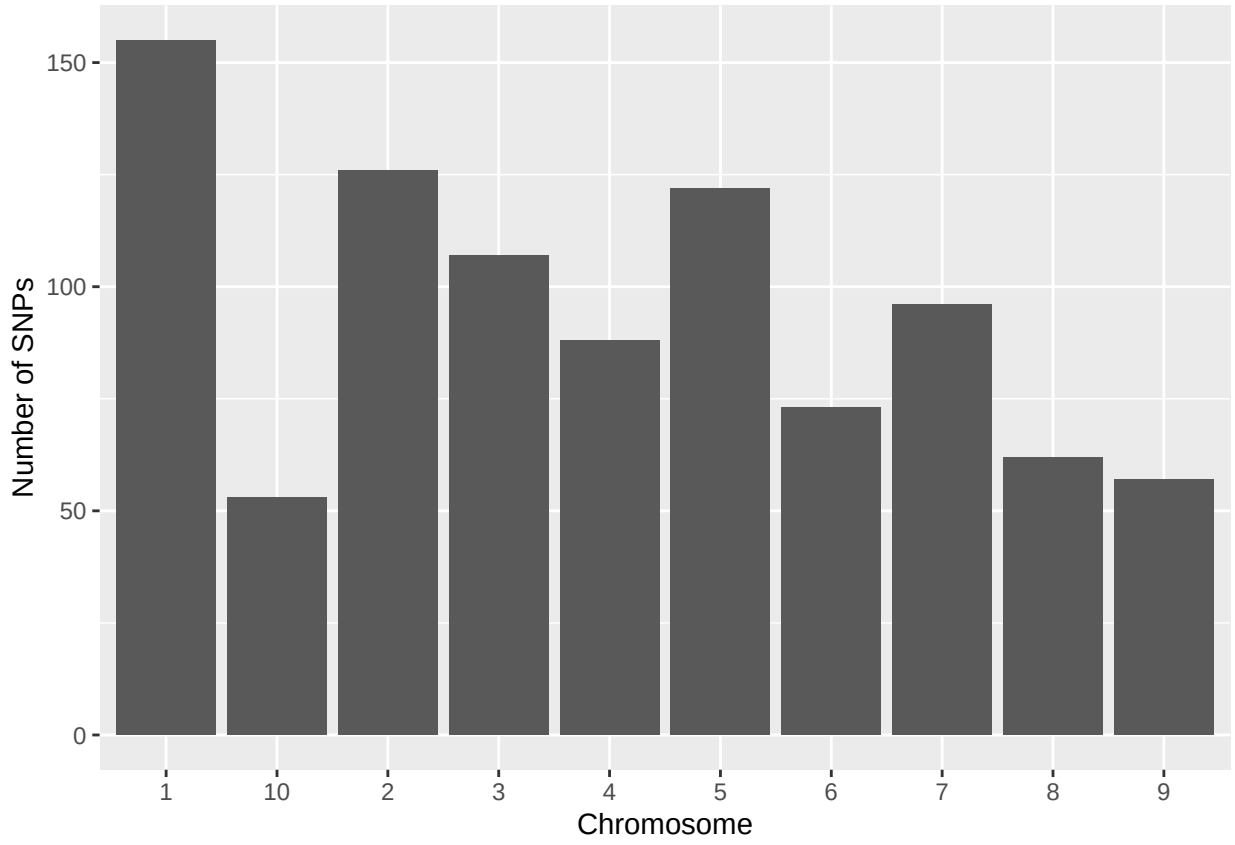
1. What is the distribution of SNPs on and accross chromosomes?

- Count the SNPs present in both Maize and Teosinte groups in each chromosome

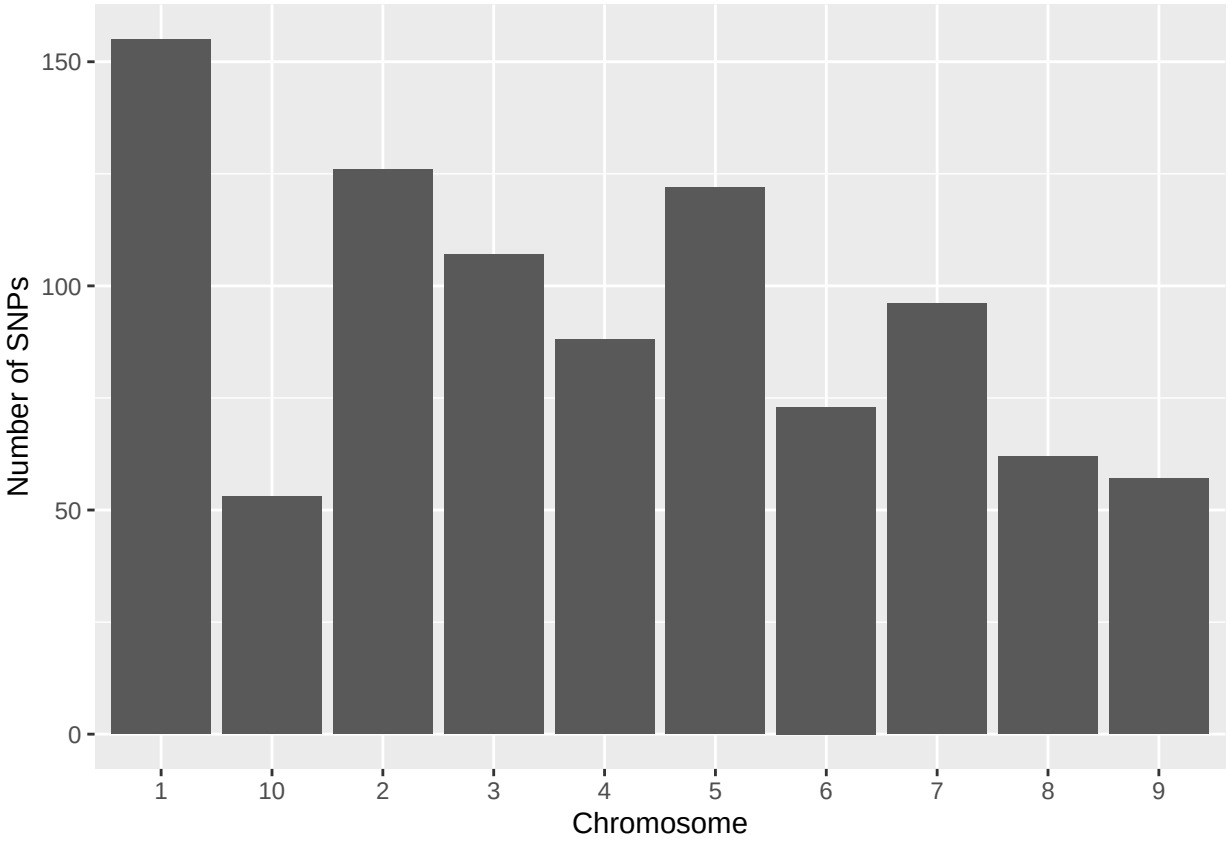
```
zmm_counts <- num_open_merged_zmm %>% count(Chromosome, sort = TRUE)
zmp_counts <- num_open_merged_zmp %>% count(Chromosome, sort = TRUE)
```

Now, create the distribution plot.

```
# Maize
ggplot() + geom_col(data = zmm_counts, mapping = aes(x=Chromosome, y=n)) + labs(x = "Chromosome", y = "
```




```
# Teosinte
ggplot() + geom_col(data = zmp_counts, mapping = aes(x=Chromosome, y=n)) + labs(x = "Chromosome", y = "
```



2. Are there more SNP positions in maize or teosinte individuals?

Visually, both Maize and Teosinte groups contain the same amount of SNPs in their chromosomes.

Missing data and amount of heterozygosity

1. What is the proportion of homozygous and heterozygous sites as well as missing data in each sample and each group?

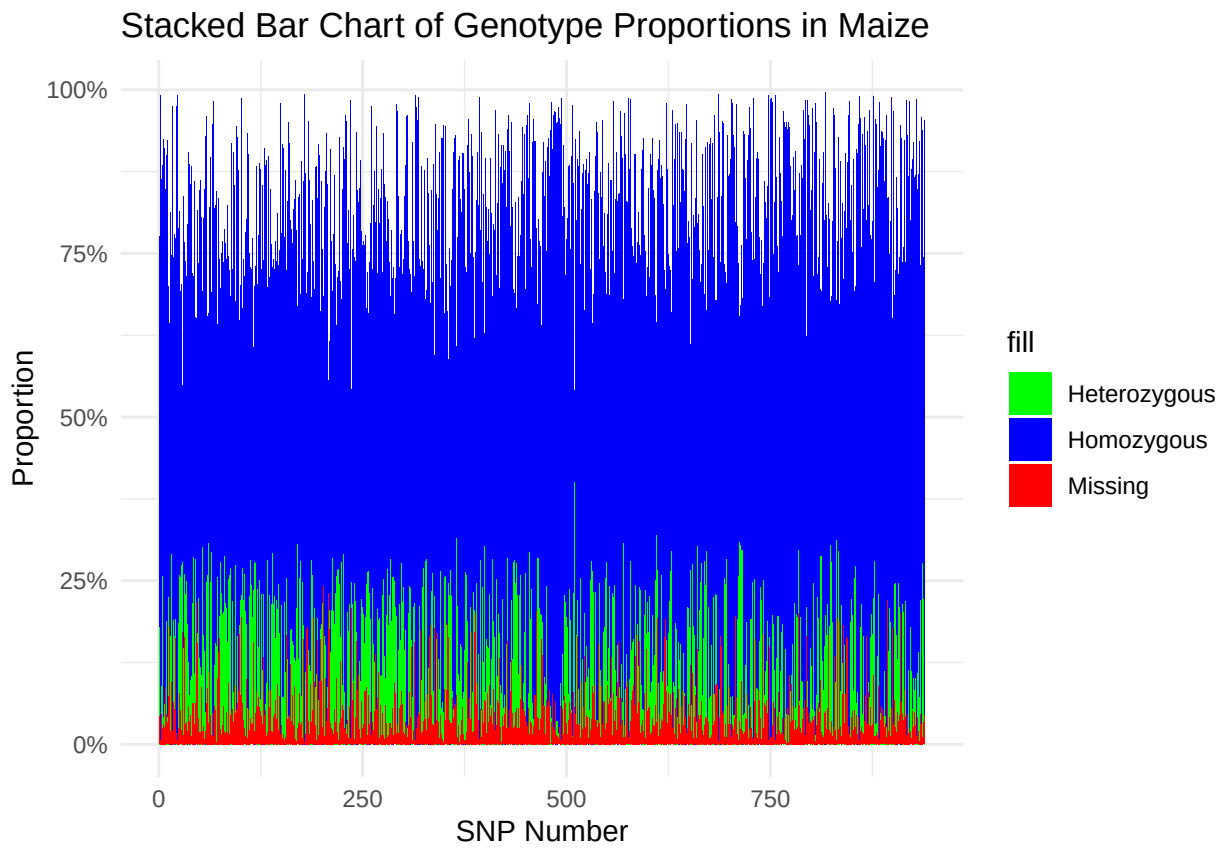
To answer this question, first we need to search and count both homozygous and heterozygous sites in our files. We will create a new data frame for those sites, as well as for the missing data.

```
# Calculate total counts for each genotype category
total_homo_zmm <- rowSums(num_open_merged_zmm == "A/A" | num_open_merged_zmm == "T/T" | num_open_merged_

total_hetero_zmm <- rowSums(num_open_merged_zmm == "A/T" | num_open_merged_zmm == "A/C" | num_open_merg

total_missing_zmm <- rowSums(num_open_merged_zmm == "?/?" | is.na(num_open_merged_zmm))

total_homo_zmp <- rowSums(num_open_merged_zmp == "A/A" | num_open_merged_zmp == "T/T" | num_open_merged_

total_hetero_zmp <- rowSums(num_open_merged_zmp == "A/T" | num_open_merged_zmp == "A/C" | num_open_merg

total_missing_zmp <- rowSums(num_open_merged_zmp == "?/?" | is.na(num_open_merged_zmp))
```

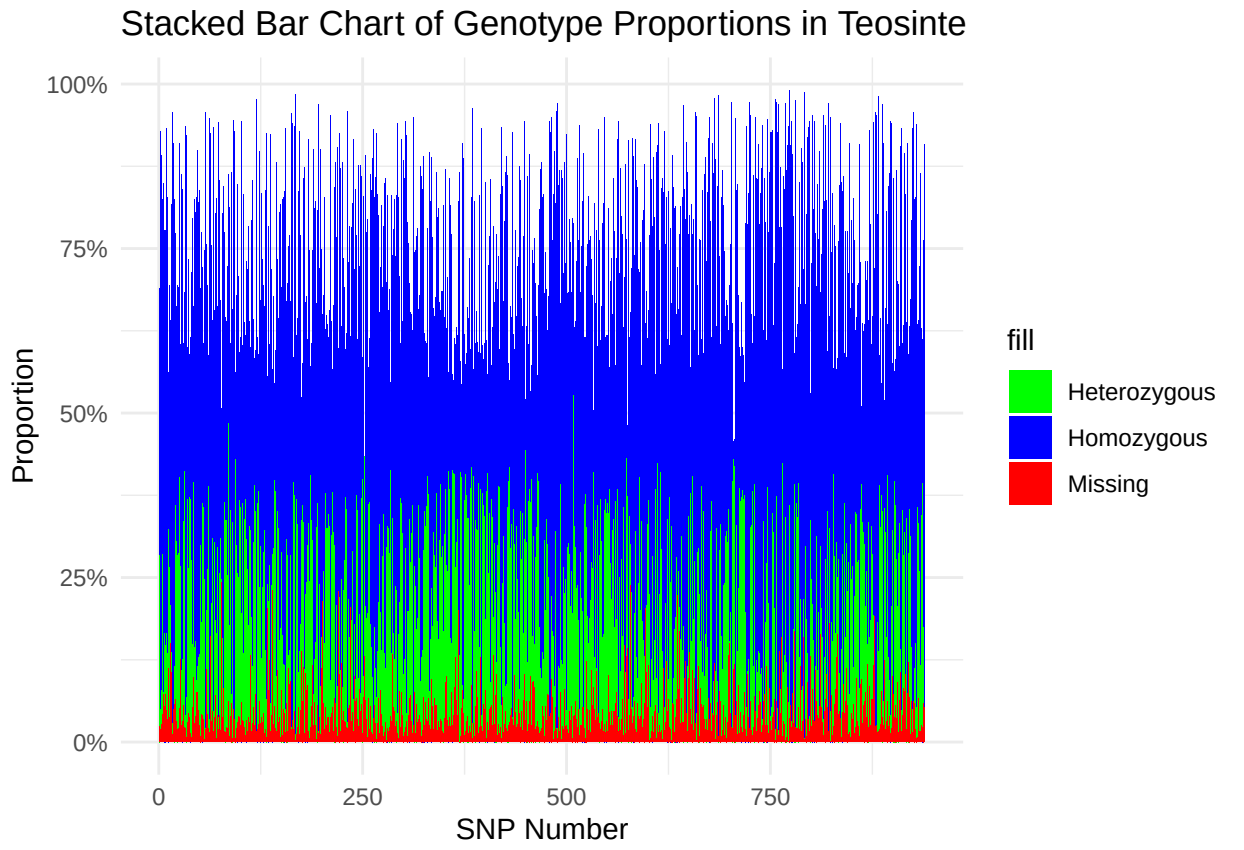
```r
# Calculate proportions
prop_homo_zmm <- total_homo_zmm / ncol(num_open_merged_zmm)
prop_hetero_zmm <- total_hetero_zmm / ncol(num_open_merged_zmm)
prop_missing_zmm <- total_missing_zmm / ncol(num_open_merged_zmm)

prop_homo_zmp <- total_homo_zmp / ncol(num_open_merged_zmp)
prop_hetero_zmp <- total_hetero_zmp / ncol(num_open_merged_zmp)
prop_missing_zmp <- total_missing_zmp / ncol(num_open_merged_zmp)

# Create data frames for proportions
zmm_proportion <- data.frame(SNP_number = 1:939, homozygous = total_homo_zmm, heterozygous = total_hete
zmp_proportion <- data.frame(SNP_number = 1:939, homozygous = total_homo_zmp, heterozygous = total_hete
```

Now, graphic the homozygous, heterozygous, missing and proportion of sites

```r
# For Maize
ggplot(zmm_proportion, aes(x = SNP_number)) +
  geom_bar(aes(y = prop_homo, fill = "Homozygous"), position = "stack", stat = "identity") +
  geom_bar(aes(y = prop_hetero, fill = "Heterozygous"), position = "stack", stat = "identity") +
  geom_bar(aes(y = prop_missing, fill = "Missing"), position = "stack", stat = "identity") +
  labs(title = "Stacked Bar Chart of Genotype Proportions in Maize",
       x = "SNP Number", y = "Proportion") +
  scale_fill_manual(values = c("Homozygous" = "blue", "Heterozygous" = "green", "Missing" = "red")) +
  scale_y_continuous(labels = scales::percent_format()) +
  theme_minimal()
```



```r
# For Teosinte
ggplot(zmp_proportion, aes(x = SNP_number)) +
  geom_bar(aes(y = prop_homo, fill = "Homozygous"), position = "stack", stat = "identity") +
  geom_bar(aes(y = prop_hetero, fill = "Heterozygous"), position = "stack", stat = "identity") +
  geom_bar(aes(y = prop_missing, fill = "Missing"), position = "stack", stat = "identity") +
  labs(title = "Stacked Bar Chart of Genotype Proportions in Teosinte",
       x = "SNP Number", y = "Proportion") +
  scale_fill_manual(values = c("Homozygous" = "blue", "Heterozygous" = "green", "Missing" = "red")) +
  scale_y_continuous(labels = scales::percent_format()) +
  theme_minimal()
```



My visualization

For my own visualization I would like to know if there are repeated SNPs in the different positions. For that, I will look at the presence of duplicate.

```r
# Check for duplicated SNP numbers in Maize data
duplicated_snps_zmm <- zmm_proportion$SNP_number[duplicated(zmm_proportion$SNP_number)]

# Check for duplicated SNP numbers in Teosinte data
duplicated_snps_zmp <- zmp_proportion$SNP_number[duplicated(zmp_proportion$SNP_number)]
```

The result is an empty integer, meaning there are no duplicated values. Thus, I will take a look at unique values instead.

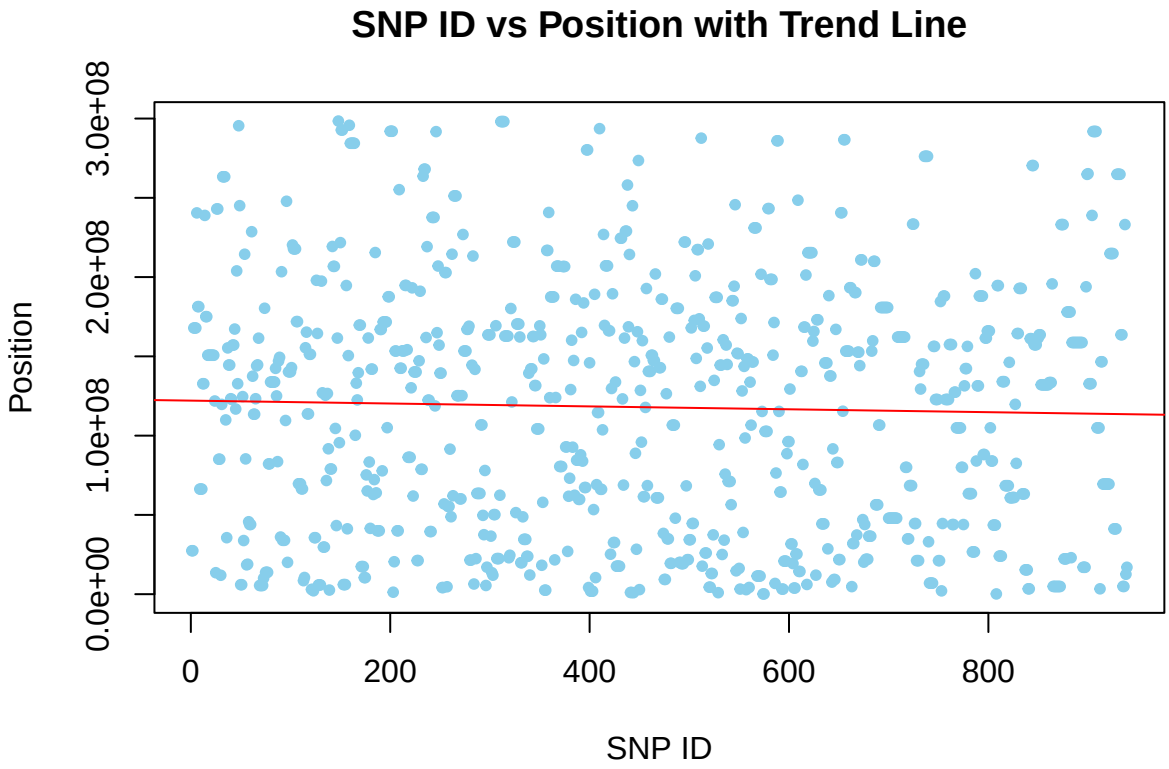
```r
# Create a data frame for the unique values in Maize
unique_snps_df <- as.data.frame(table(zmm_proportion$SNP_number))

# Rename the columns for clarity
colnames(unique_snps_df) <- c("SNP_Number", "Frequency")

# Extract chromosome information
chromosome <- num_open_merged_zmm[, 2]

# Create a scatter plot
plot(unique_snps_df$SNP_Number, chromosome,
     pch = 20, col = "skyblue",
     main = "SNP ID vs Chromosome",
     xlab = "SNP ID", ylab = "Chromosome")
```

* When plotting this, I found out that every SNP ID from the dataset is unique (they are not duplicated).

- Something else I can plot is the distribution of the SNPs according to their position.I will add a linear regression trend line to see if there is any trend amon SNP ID related to their position.

```r
# Extract position information
snp_positions <- num_open_merged_zmm[, 3]

# Create a data frame with SNP ID and position columns
snp_data <- data.frame(SNP_ID = seq_along(snp_positions), Position = snp_positions)
```

```r
# Fit linear regression model
lm_model <- lm(Position ~ SNP_ID, data = snp_data)

# Extract coefficients
coefficients <- coef(lm_model)

# Plot SNP ID versus position with the trend line
plot(snp_data$SNP_ID, snp_data$Position,
     pch = 20, col = "skyblue",
     main = "SNP ID vs Position with Trend Line",
     xlab = "SNP ID", ylab = "Position")
abline(coefficients, col = "red")
```



The line is very much flat, which suggests there is not a trend beteen the SNP ID and their positions.