#UNIX Assignment

##Data Inspection

###Attributes of fang\_et\_al\_genotypes

Here is my snippet of code used for data inspection

wc fang\_et\_al\_genotypes.tx

wc -l fang\_et\_al\_genotypes.tx #gave me only the lines 2783

du -h fang\_et\_al\_genotypes.txt

awk -F "\t" '{print NF; exit}' fang\_et\_al\_genotypes.txt

By inspecting this file, I learned:

1. The .txt file contains 2783 lines, 2744038 words, and 11051939 number of bytes in the files.
2. This gave me only the number of lines is 2783
3. File size is 6.7M
4. Number of columns is 986

###Attributes of snp\_position.txt

Here is my snipped of code used for data inspection

wc snp\_position.txt

wc -l snp\_position.txt

du -h snp\_position.txt

awk -F "\t" '{print NF; exit}' snp\_position.txt

By inspecting this file, I learned:

1. The snp\_position.txt file contains 984 lines, 13198 words, and 82763 number of bytes in the files.
2. This gave me only the number of lines is 984
3. File size is 49K
4. Number of columns is 15

###Data processing

###Maize Data

Here is my snippet of code used for data processing

1. tail -n +3 transposed\_genotypes.txt > Transposed\_1header.txt
2. sed 's/Group/SNP\_ID1/g' Transposed\_1header.txt > transposed\_Group.txt
3. tail -n +6 Transposed\_1header.txt | awk -F "\t" '{print NF; exit}'
4. tail -n +6 transposed\_Group.txt | awk -F "\t" '{print NF; exit}'
5. join -1 1 -2 1 -t $'\t' snp\_position.txt transposed\_Group.txt > transjoin.txt
6. awk '{for(i=1;i<=NF;i++) if ($i == "ZMMIL") print "Found at column " i}' transjoin.txt
7. awk '{for(i=1;i<=NF;i++) if ($i == "ZMPJA") print "Found at column " i}' transjoin.txt
8. awk '{for(i=1;i<=NF;i++) if ($i == "ZMMMR") print "Found at column " i}' transjoin.txt
9. awk '{for(i=1;i<=NF;i++) if($i == "ZMMMR") count++} END{print count}' transjoin.txt
10. awk '{for(i=1;i<=NF;i++) if($i == "ZMMLR") count++} END{print count}' transjoin.txt
11. awk '{for(i=1;i<=NF;i++) if($i == "ZMMIL") count++} END{print count}' transjoin.txt
12. cut -d $'\t' -f1,3,4,2508-2797,1225-2480,2481-2507 transjoin.txt > maize.txt
13. (head -n 1 maize.txt && tail -n +2 maize.txt | sort -k3,3n) > maize\_sorted.txt
14. grep -v "^#" maize\_sorted.txt | cut -f2 | sort | uniq -c
15. mkdir Maize\_Increasing Maize\_Decreasing
16. for i in {1..10}; do

{ head -n 1 updated\_maize\_sorted.txt && awk -v chr="$i" '$2 == chr' updated\_Sortedmaize\_file.txt; } > "Maize\_chromo${i}.txt"

mv "Maize\_chromo${i}.txt" Maize\_Increasing/

done

1. { head -n 1 maize\_sorted.txt && awk '$2 == "multiple"' maize\_sorted.txt; } > Maize\_chrM.txt && { head -n 1 maize\_sorted.txt && awk '$2 == "unknown"' maize\_sorted.txt; } > Maize\_chrU.txt
2. Mv Maize\_chrU.txt Maize\_ChrM.txt Maize\_Increasing/
3. sed 's/?/-/g' maize\_sorted.txt > maize-sorted.txt
4. mv maize-sorted.txt Maize\_Decreasing
5. cd Maize\_Decreasing
6. (head -n 1 maize-sorted.txt && tail -n +2 maize-sorted.txt | sort -k3,3nr) > MDsorted.txt
7. out\_file="Maize\_chromo\_Decreasing\_sorted${i}.txt"

    { head -n 1 MDsorted.txt && awk -v chr="$i" '$2 == chr' MDsorted.txt; } > "$out\_file"

   echo "Chromosome $i: $(wc -l < "$out\_file") lines"

Here is my brief description of what this code does

1. Tail -n +3 print the line 3 onward instead of printing the default last few lines. It skipped the whole row of Sample\_ID and JG\_OTU, redirected the output of the tail command of transposed\_genotypes.txt into new file called Transposed\_1header.txt.
2. stream editor (sed) substitute all “Group” with “SNP\_ID” globally in Transposed\_1header.txt and redirect the modified output to transposed\_Group.txt
3. tail -n +3 skips the first 5 lines and output the rest of the file, -F”\t” tells awk to use tab(\t) as the column separator and print the number of fields (column).
4. Same as 3.
5. Join command merge two files based on their first column (SNP ID) of the first line using the tab delimiter and saved the output in transjoin.txt
6. Awk read each line, for(i=1;i<=NF;i++) goes through each column in the current row of the file until i=1. If it finds “ZIMML” in any of the column, it printed as “Found at column” followed by the column number i.
7. Awk read each line, for(i=1;i<=NF;i++) goes through each column in the current row of the file until i=1. If it finds “ZMPJA” in any of the column, it printed as “Found at column” followed by the column number i.
8. Awk read each line, for(i=1;i<=NF;i++) goes through each column in the current row of the file until i=1. If it finds “ZMMR” in any of the column, it printed as “Found at column” followed by the column number i.
9. It prints the final count of how many times “ZMMMR” appeared across all rows in the file.
10. Same as 9 searches for “ZMMLR”.
11. Same as 9 searches for “ZMMIL”.
12. Cut command select extract columns (f1,3,4,2508-2797,1225-2480,2481-2507) and saved into a new file called maize.txt.
13. It extracts the first line the header, && ensure second part execute only when header is extracted. Tail -n +2 extracts all lines except the first from maize.txt. The rest of the files excluding the first line pipes it into sort -k3, 3n. It then sorts data numerically (n) by the third column (k3,3) in ascending order. It redirects the combined output (header + sorted data) into a new file called maize\_sorted.txt.
14. Excludes the lines that start with #, extract lines from second column, sort in ascending order, count how many times each unique values appear in the second column.
15. Makes two working directory
16. Loops from i=1 to i=10, extract the first line from the line, (awk -v chr=”I”) sets the chr variable in awk to the current value of i ($2 == chr’) check if value in second column is equal to chr (current value of i). The filtered rows (awk going line by line) along with header are saved in Maize\_chromo{i}.txt. All files are then moved into Maize\_Increasing directory.
17. Extract the header and filter rows based on multiple and unknown from column 2.
18. Move both files into working directory.
19. t reads the maize\_sorted.txt file, replaces all “?” with “-“globally and saved into maize-sortex.txt.
20. Move maize-sorted file into Maize\_Decreasing
21. Changing directory
22. Same as 13 but sort in decreasing order from MDsorted.txt file.
23. For each chromosome ($i from 1 to 10), the script filters the sorted data (MDsorted.txt) to include any rows where the second column matches the chromosomes number and writes the result into new file named Maize\_Decreasing\_sorted{i}.txt. echo command prints the number of lines in each new file using the wc -l command, which count the number of lines in the file

###Teosinte Data

1. awk '{for(i=1;i<=NF;i++) if ($i == "ZMPBA") print "Found at column " i}' transjoin.txt
2. awk '{for(i=1;i<=NF;i++) if ($i == "ZMPJA") print "Found at column " i}' transjoin.txt
3. awk '{for(i=1;i<=NF;i++) if ($i == "ZMPIL") print "Found at column " i}' transjoin.txt
4. cut -d $'\t' -f1,3,4,89-988,1178-1218,989-1022 transjoin.txt > Teosinte.txt
5. (head -n 1 Teosinte.txt && tail -n +2 Teosinte.txt | sort -k3,3n) > Teosinte\_sorted.txt
6. grep -v "^#" Teosinte\_sorted.txt | cut -f2 | sort | uniq -c
7. { head -n 1 Teosinte\_sorted.txt && awk -v chr="$i" '$2 == chr' Teosinte\_sorted.txt; } > "Teosinte\_chromo${i}.txt"
8. mv Teosinte\_chromo\*.txt Teosinte/
9. { head -n 1 Teosinte\_sorted.txt && awk '$2 == "multiple"' Teosinte\_sorted.txt; } > Teosinte\_chromoM.txt && \

{ head -n 1 Teosinte\_sorted.txt && awk '$2 == "unknown"' Teosinte\_sorted.txt; } > Teosinte\_chromoU.txt

1. mv Teosinte\_chromoM.txt Teosinte\_chromoU.txt Teosinte/
2. sed 's/?/-/g' Teosinte\_sorted.txt > Teosinte-sorted.txt
3. for i in {1..10}; do

{ head -n 1 TDsorted && awk -v chr="$i" '$2 == chr' TDsorted; } > "Teosinte\_chromo${i}.txt"

Done

Here is my brief description of what this code does

1. Search for the string “ZMPBA” in each column of transjoin.txt and print the no of column.
2. Same as 1 search for “ZMPJA”
3. Same as 2, search for “ZMPIL”
4. Extracts columns 1, 3, 4, and ranges 89-988, 1178-1218, 989-1022 from transjoin.txt and saves it as Teosinte.txt.
5. Sorts the Teosinte.txt file by the 3rd column in ascending numerical order and saves it as Teosinte\_sorted.txt. The header is retained at the top.
6. Removes lines starting with #, extracts the 2nd column, sorts, and counts the unique occurrences of each value.
7. For each chromosome (1-10), this extracts the lines where the 2nd column equals the chromosome number and saves it into separate files like Teosinte\_chromo1.txt, Teosinte\_chromo2.txt, etc.
8. Moves all Teosinte\_chromo\*.txt files into the Teosinte/ directory.
9. Extracts rows where the 2nd column is “multiple” and “unknown” and saves them in Teosinte\_chromoM.txt and Teosinte\_chromoU.txt.
10. Moves the files Teosinte\_chromoM.txt and Teosinte\_chromoU.txt into the Teosinte/ directory.
11. Replaces all occurrences of ? with - in Teosinte\_sorted.txt and saves the result as Teosinte-sorted.txt.
12. For chromosomes 1 to 10, this filters rows based on the 2nd column matching the chromosome number and saves them as separate files (Teosinte\_chromo1.txt, Teosinte\_chromo2.txt, etc.).