

## ADM analysis on 10,000 shuffled datasets

There are 7 steps to perform shuffling, mixscore ADM analysis, and data analysis.

**Step 1:** Read in raw SNP blocks data from 22 chromosome-based datasets; Perform circle-shuffling, save shuffled data to txt documents. This step will create 1000 shuffled datasets, as txt documents plus the original unshuffled dataset.

**Step 2:** Generate 1001 parameter files, for mixscore software to use. Have to put these parameter files in a folder in DSCR.

**Step 3:** Write 1000 Sbatch scripts, to allocate CPUs and Memory for each job to use. These sbatch jobs has to be put inside mixscore software package, in our case: /work/AndrewGroup/ADM\_mixscore\_test/mixscore/

**Step 4:** Use Java run-time to submit 1001 jobs directly to Duke Cluster (DSCR). This step would take up to 5 days, depends on how busy the cluster would be, some jobs would be rejected and have to reboot manually. In the end, 1001 mixscore results would be generated as txt documents, in each document, there is one vector of 229860 items.

**Step 5:** Merge 1001 mixscore results into one huge matrix (229860 \* 1001). Save the matrix to a txt document.

**Step 6:** Get the maximum value from each column in the huge matrix.

REPEATED step 1 through step 6 for 10 times, thus we got 10,000 shuffled datasets' mixscore ADM results.

**Step 7:** Extract first 10 maximum values from each column of each matrix we got after step 6.

**Step 8:** Perform simulation and calculate p-value/percentage for each of the 10 max values.

**Results:**

```
# > max.unshuffled  
  
# the first 10 maximum values from unshuffled dataset's ADM analysis result.  
# [1] 16.6244 15.9254 15.2406 14.5702 13.9141 13.2725 13.0877 12.6453  
12.3885 12.0326  
  
### Print out p-value for the first 10 maxium values in the unshuffled dataset:  
# [1] 1st maximum value percentage: 0.0179  
# [1] 2nd maximum value percentage: 0.0286  
# [1] 3rd maximum value percentage: 0.0455  
# [1] 4th maximum value percentage: 0.0532  
# [1] 5th maximum value percentage: 0.0773  
# [1] 6th maximum value percentage: 0.1129  
# [1] 7th maximum value percentage: 0.1297  
# [1] 8th maximum value percentage: 0.1575  
# [1] 9th maximum value percentage: 0.1812  
# [1] 10th maximum value percentage: 0.2135  
  
>
```

So, here we could see, the 5th maximum value is not significant.

Protocol in detail

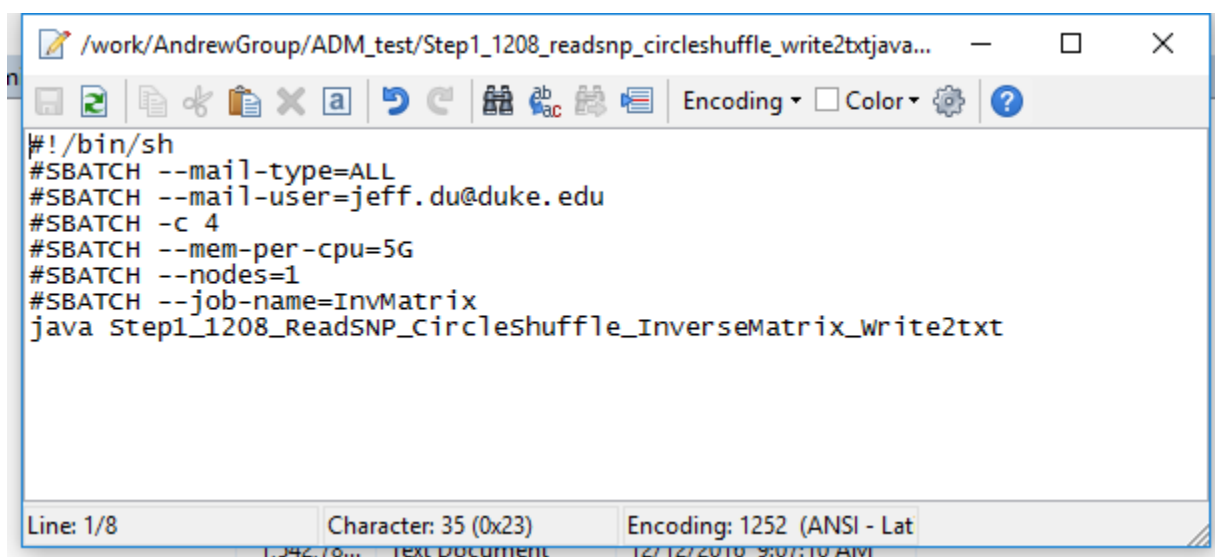
Step 1: read in raw data, perform circle shuffle, generate outputs.

Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step1\\_0102\\_ReadSNP\\_CircleShuffle\\_InverseMatrix\\_Write2txt.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step1_0102_ReadSNP_CircleShuffle_InverseMatrix_Write2txt.java)

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/SNP\\_Block.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/SNP_Block.java)

Sbatch script sample:

A screenshot of a text editor window. The title bar shows the file path: /work/AndrewGroup/ADM\_test/Step1\_1208\_readsnp\_circlesuffle\_write2txtjava... The editor contains an sbatch script. The script starts with a shebang line, followed by several SBATCH directives for email, user, cores, memory, nodes, and job name, and ends with a java command. The status bar at the bottom shows 'Line: 1/8', 'Character: 35 (0x23)', and 'Encoding: 1252 (ANSI - Lat)'.

```
#!/bin/sh
#SBATCH --mail-type=ALL
#SBATCH --mail-user=jeff.du@duke.edu
#SBATCH -c 4
#SBATCH --mem-per-cpu=5G
#SBATCH --nodes=1
#SBATCH --job-name=InvMatrix
java Step1_1208_ReadSNP_Circlesuffle_InverseMatrix_Write2txt
```

Data input:

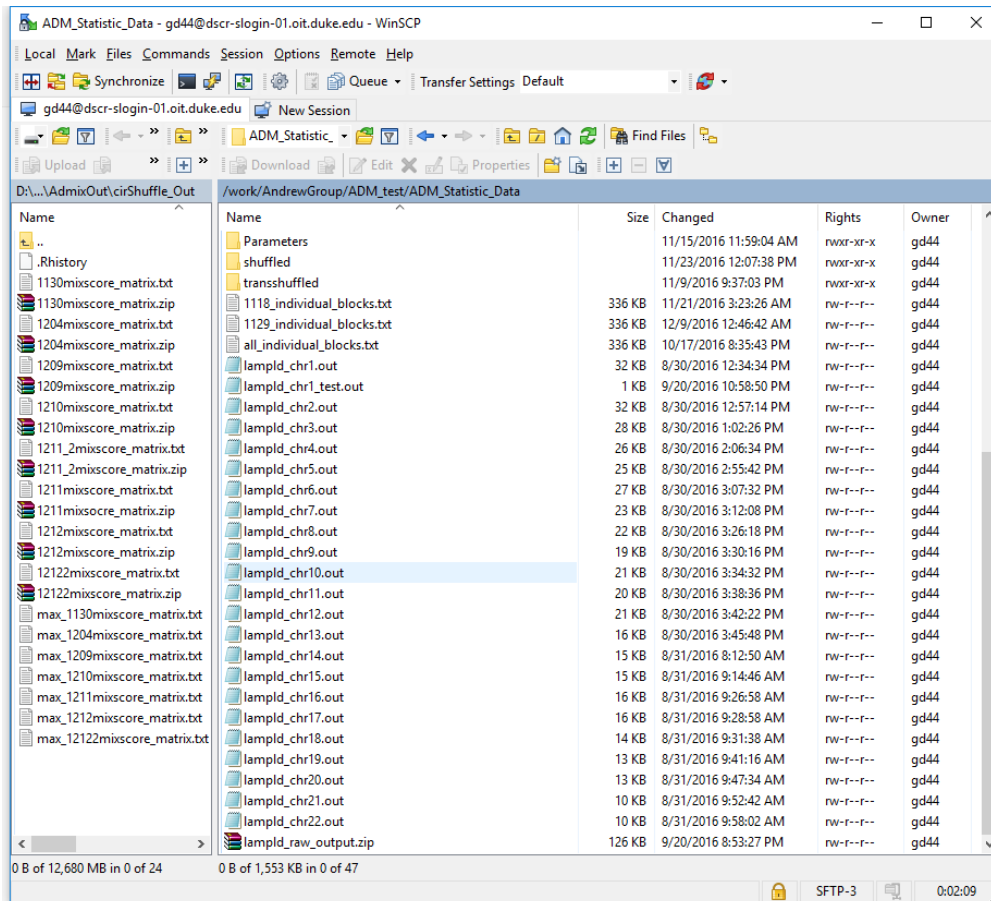
Lampld\_chr1.out

Lampld\_chr2.out

...

...

Lampld\_chr22.out



Data output:

1matrix.txt

2matrix.txt

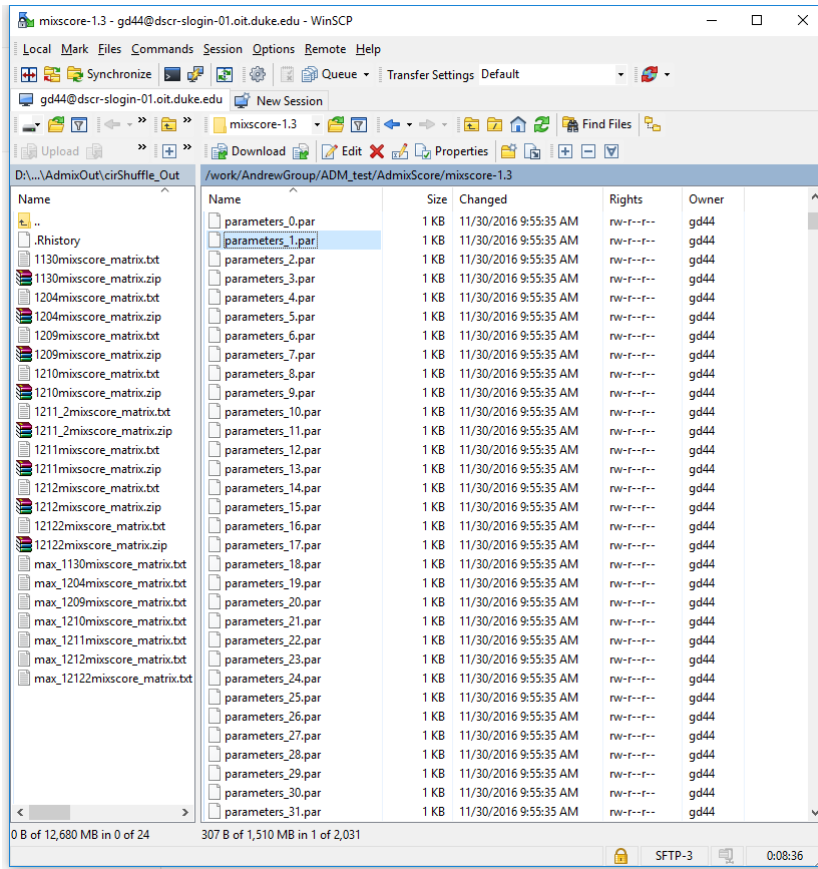
...

...

1000matrix.txt

After running the Step #1, we must change the name of the folder where we put these 1000 outputs.





Step 3: write Sbatch scripts for jobs submitting to DSCR.

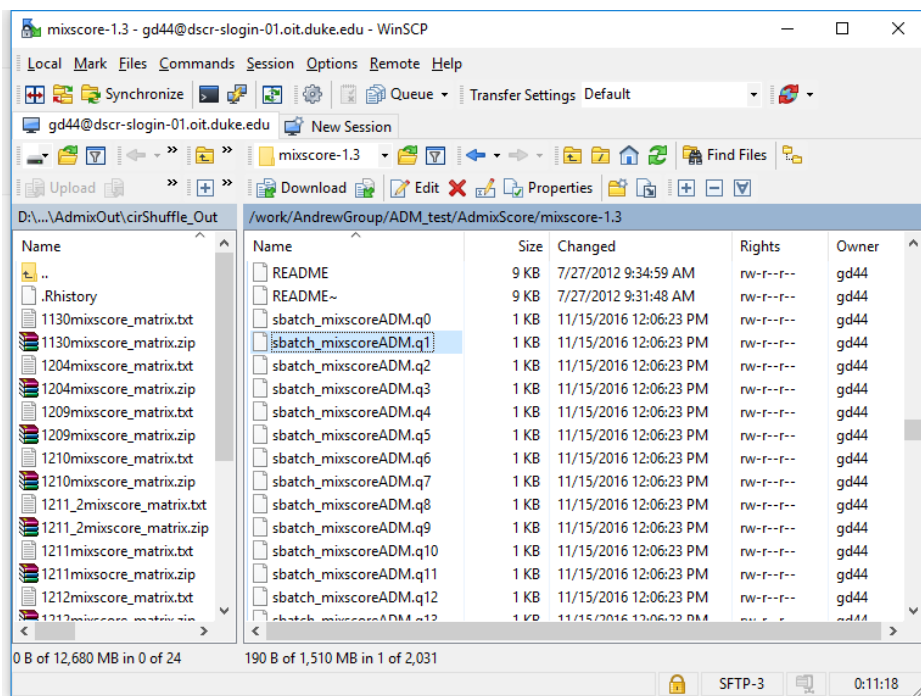
Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step3\\_0102\\_write\\_Sbatch\\_scripts.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step3_0102_write_Sbatch_scripts.java)

One sample of Sbatch scripts:

```
/work/AndrewGroup/ADM_test/AdmixScore/mixscore-1.3/sbatch_mixscoreADM.q1 - gd44@ds...  
#!/bin/sh  
#SBATCH --mail-type=ALL  
#SBATCH --mail-user=todedo@gmail.com  
#SBATCH -c 2  
#SBATCH --mem-per-cpu=5G  
#SBATCH --nodes=1  
#SBATCH --job-name=mixs1th  
./bin/mixscore ADM parameters_1.par  
Line: 1/8 Column: 1 Character: 35 (0x23) Encoding: 1252 (ANSI - Lat
```

In total, there are 1001 sbatch scripts.

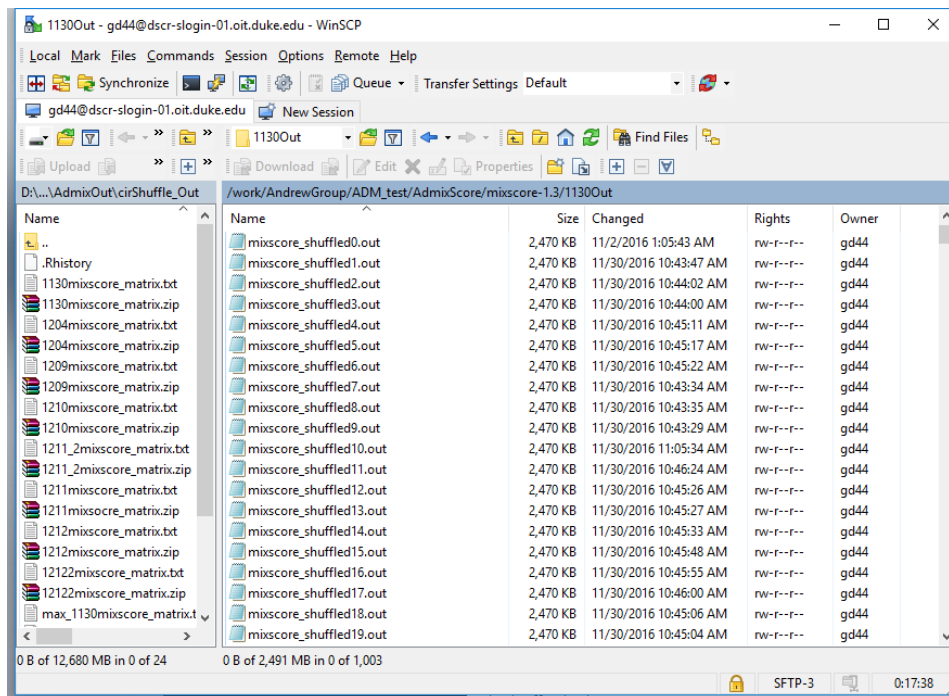


Step 4: submit 1000 jobs to Duke Cluster.

Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step4\\_0102\\_Submit\\_1000\\_sbatch\\_cmd.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step4_0102_Submit_1000_sbatch_cmd.java)

After submitting 1001 jobs to DSCR, it might take a few days to get all 1001 results.



Put all these 1001 \*.out outputs in one folder.

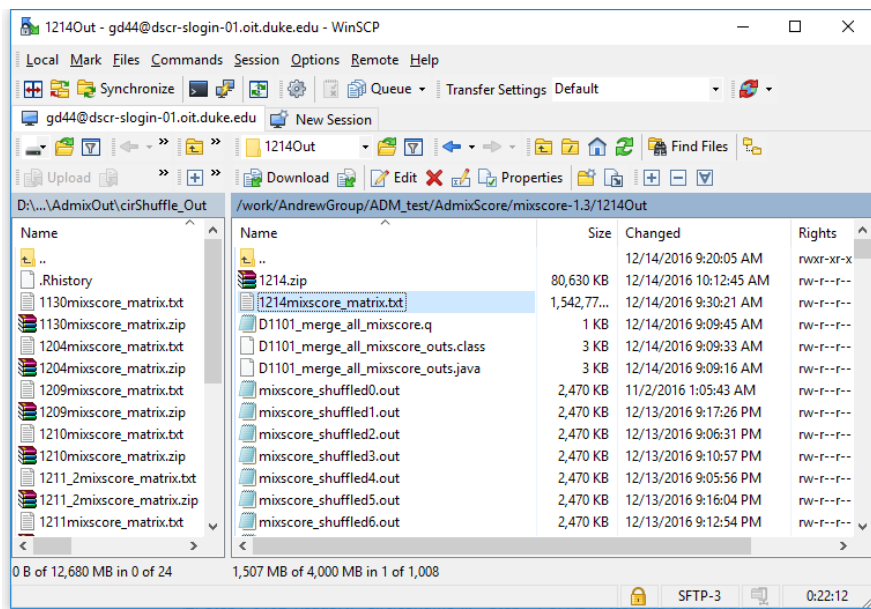
Step 5: merge all 1001 mixscore outputs into one matrix.

Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step5\\_0102\\_merge\\_all\\_mixscore\\_outs.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step5_0102_merge_all_mixscore_outs.java)

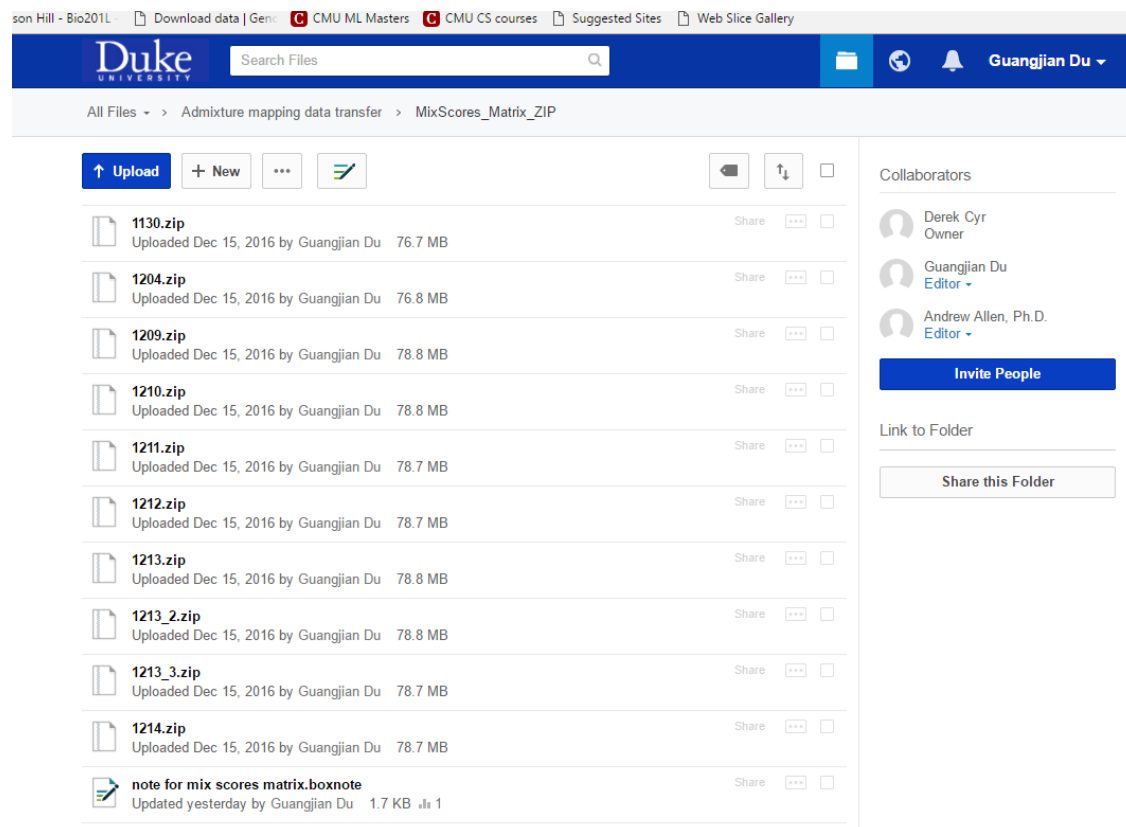
As shown in the figure below:





Download the merged matrix document to local desktop.

I have put all these results to Duke Box.



Step 6: get the maximum value from each column in the giant matrix.

Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step6\\_0102\\_extra\\_ct\\_Max\\_ADM\\_results.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step6_0102_extra_ct_Max_ADM_results.java)

Here, we only got 1001 mixscore ADM results, so we repeated step 1 – step 6 for 10 times, and got 10010 mixscore ADM results.

These results have been put to Duke box too.

The screenshot shows the Duke University file sharing interface. The top navigation bar includes the Duke University logo, a search bar, and user information for 'Guangjian Du'. The breadcrumb trail indicates the current location: 'All Files > Admixture mapping data transfer > 1stMaxValues'. The main content area displays a list of files and folders. On the right side, there are sections for 'Collaborators' (listing Derek Cyr as Owner, Guangjian Du as Editor, and Andrew Allen, Ph.D. as Editor) and 'Link to Folder' (with a 'Share this Folder' button).

| File Name                       | Uploaded by  | Size   |
|---------------------------------|--------------|--------|
| max_mixscore_1.txt              | Guangjian Du | 7.2 KB |
| max_mixscore_10.txt             | Guangjian Du | 7.2 KB |
| max_mixscore_2.txt              | Guangjian Du | 7.2 KB |
| max_mixscore_3.txt              | Guangjian Du | 7.3 KB |
| max_mixscore_4.txt              | Guangjian Du | 7.3 KB |
| max_mixscore_5.txt              | Guangjian Du | 7.2 KB |
| max_mixscore_6.txt              | Guangjian Du | 7.2 KB |
| max_mixscore_7.txt              | Guangjian Du | 7.2 KB |
| max_mixscore_8.txt              | Guangjian Du | 7.2 KB |
| max_mixscore_9.txt              | Guangjian Du | 7.2 KB |
| note for 1st Max Values.boxnote | Guangjian Du | 1.7 KB |

Step 7: get the first 10 maximum values from each column, in the 10 gaint matrix.

Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step7\\_0102\\_extra\\_ct\\_10Max\\_ADM\\_results.java](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step7_0102_extra_ct_10Max_ADM_results.java)

Step 8: check p-value/percentage for each maximum value.

Code:

[https://github.com/breezedu/ADM\\_Statistic\\_test/blob/master/Step8\\_0102\\_check\\_10maxValue\\_percentile.R](https://github.com/breezedu/ADM_Statistic_test/blob/master/Step8_0102_check_10maxValue_percentile.R)

Results:

```
# > max.unshuffled

# the first 10 maximum values from unshuffled dataset's ADM analysis result.
# [1] 16.6244 15.9254 15.2406 14.5702 13.9141 13.2725 13.0877 12.6453
12.3885 12.0326

### Print out p-value for the first 10 maximum values in the unshuffled dataset:
# [1] 1st maximum value percentage: 0.0179
# [1] 2nd maximum value percentage: 0.0286
# [1] 3rd maximum value percentage: 0.0455
# [1] 4th maximum value percentage: 0.0532
# [1] 5th maximum value percentage: 0.0773
# [1] 6th maximum value percentage: 0.1129
# [1] 7th maximum value percentage: 0.1297
# [1] 8th maximum value percentage: 0.1575
# [1] 9th maximum value percentage: 0.1812
# [1] 10th maximum value percentage: 0.2135

>
```

So, here we could see, the 5th maximum value is not significant.

## About the circle shuffling:

Q: Why there would be two peaks in the beginning and the end:

The first and the last blocks are 'ALLIGNED', which means the proportion of 0s, 1s, and 2s are the proportions of 0-type blocks, 1-type blocks, and 2-type blocks.

In the original dataset, among all the 565 **first** blocks, the percentage of 2-type SNP-blocks is only 2%;

Over all 565 individual and over all blocks, the percentage of 2-type SNP-blocks take up ~6%;

After shuffling, the percentage of 2-type SNP-blocks in the first row of blocks would be raised to ~6, that's why we could observe a peak in the beginning.

The same scenario would happen for the very last blocks.

Compare the percentages of 0-blocks, 1-blocks, and 2-blocks, between unshuffled dataset and shuffled datasets, we could see the shuffling normalized the percentages for all blocks.

### Unshuffled

|                   | 0-blocks, | 1-blocks, | <b>2-blocks..</b> |
|-------------------|-----------|-----------|-------------------|
| The first blocks: | 58.94%    | 39.12%    | 1.95%             |
| The last blocks:  | 57.35%    | 39.65%    | 3.01%             |
| Over all blocks:  | 46.73%    | 47.03%    | 6.23%             |

### Shuffle #1 done!

|                   | 0-blocks, | 1-blocks, | <b>2-blocks..</b> |
|-------------------|-----------|-----------|-------------------|
| The first blocks: | 46.55%    | 46.37%    | 7.08%             |
| The last blocks:  | 46.90%    | 46.90%    | 6.19%             |
| Over all blocks:  | 46.73%    | 47.03%    | 6.23%             |

### Shuffle #2 done!

|                   | 0-blocks, | 1-blocks, | <b>2-blocks..</b> |
|-------------------|-----------|-----------|-------------------|
| The first blocks: | 52.39%    | 40.53%    | 7.08%             |
| The last blocks:  | 45.84%    | 47.96%    | 6.19%             |
| Over all blocks:  | 46.73%    | 47.03%    | 6.23%             |

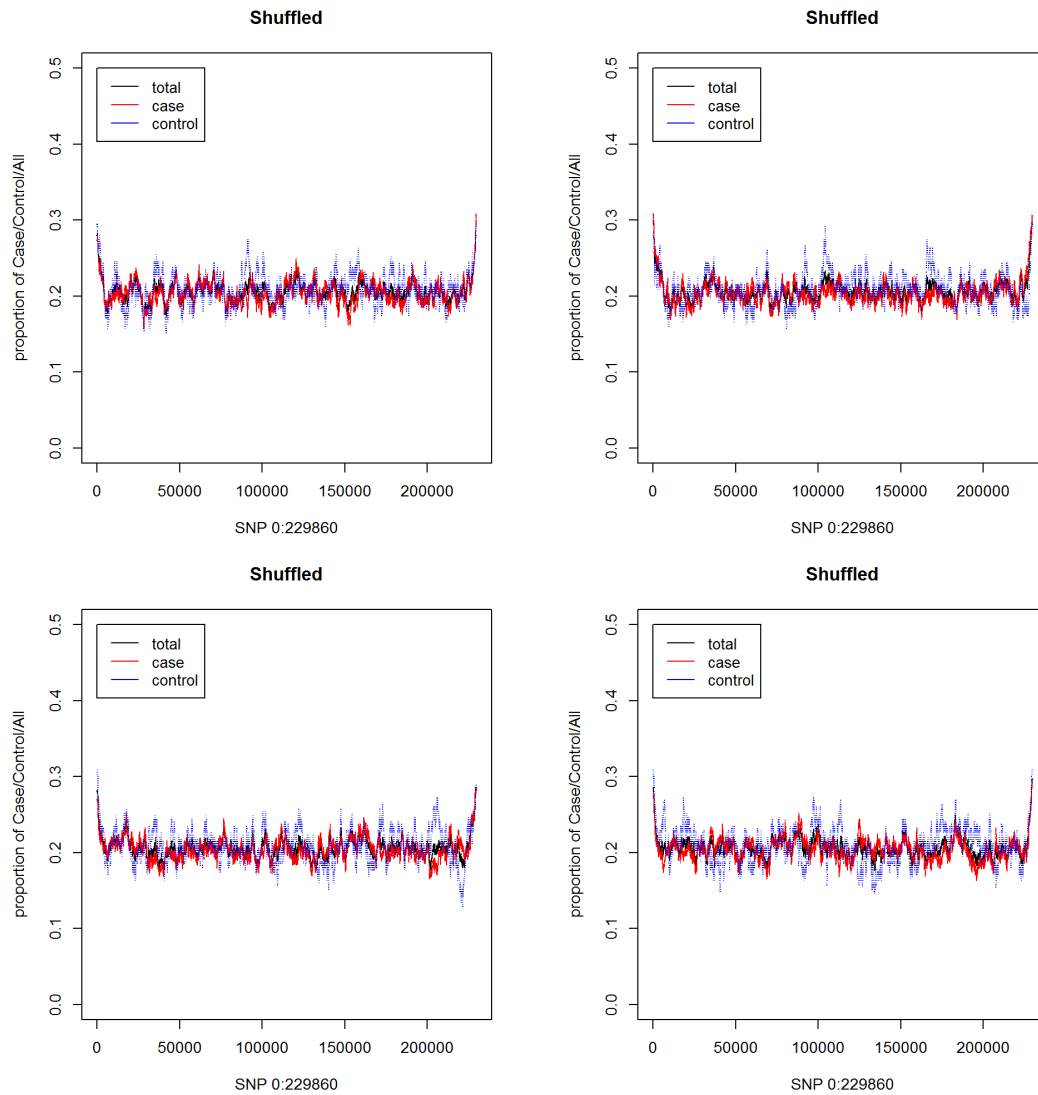


Fig 1 ancestry proportion of normally shuffled dataset

As shown in Fig 1, this is not what we expected.

Solution:

How about we link the last block with the first block for each individual, then made a 'circle' of SNP-blocks.

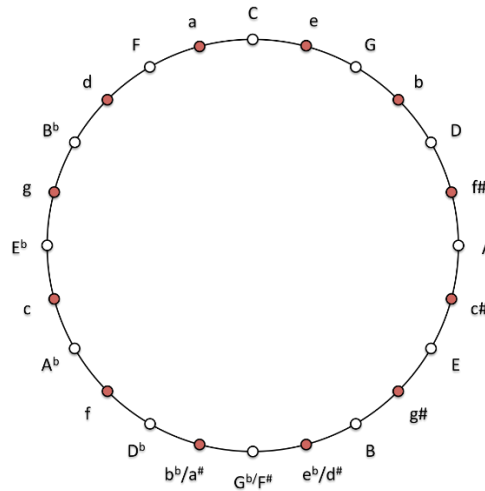


Fig 2 a circle linked blocks for one individual

After we build a circle linked SNP-blocks for a person, we shuffle it with permutation method (3x).

Align all circled SNP-blocks like this: this is a visualization of 6 individuals aligned together.



Fig 3 Align circled SNP-blocks

Then, pick a random SNP position instead of purposely pick a Block start; Open the circle, get a new array of 229860 SNPs.

Align all 565 arrays of SNPs to get a huge matrix. Still, we will get a 229860\*565 matrix.

Calculate the ancestry proportion for the unshuffled and circled shuffled datasets:

The ancestry proportion plot of first 10,000 SNPs are shown in Fig 3.

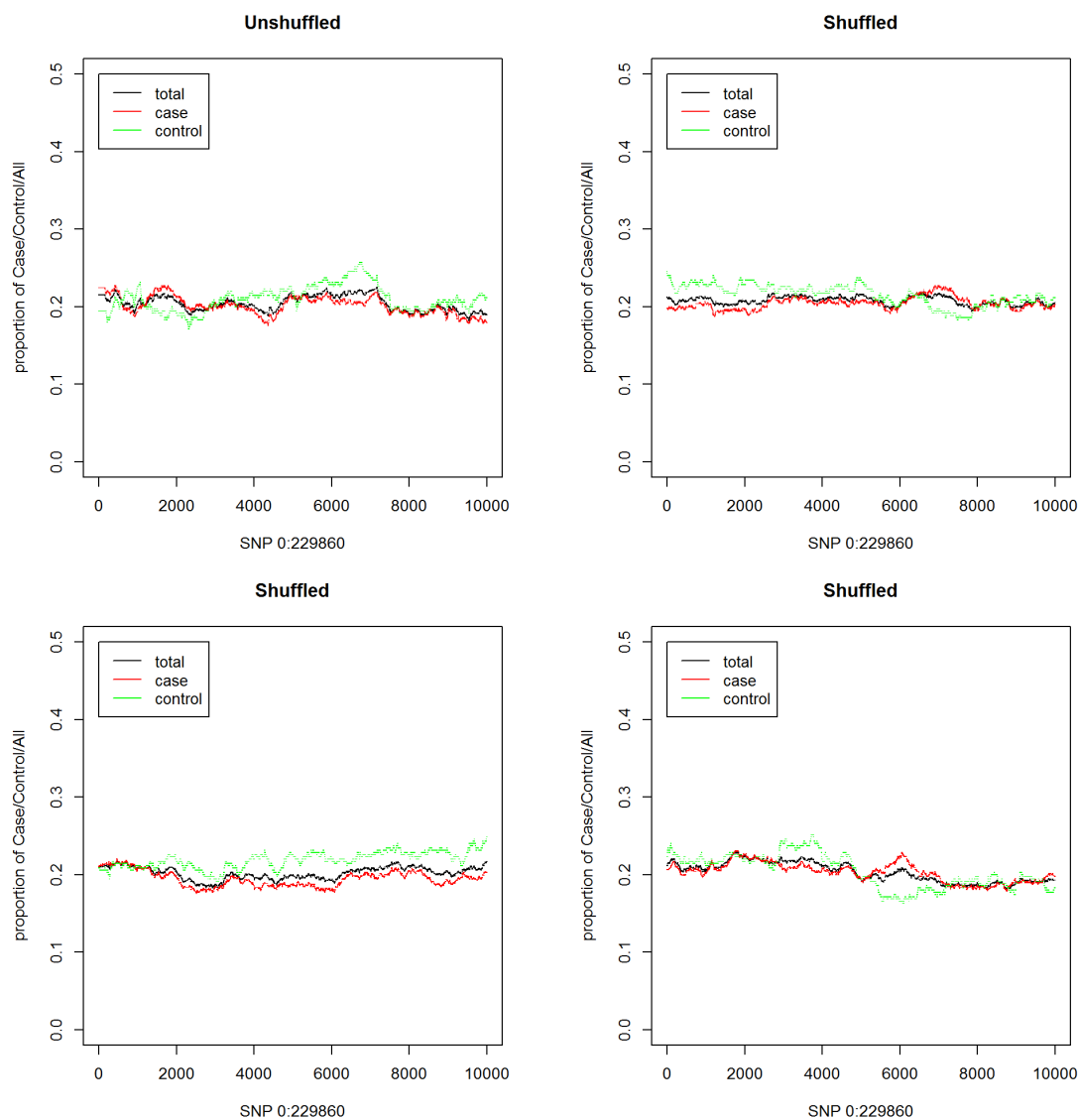


Fig 4 1-10,000 SNPs ancestry proportion plot.

The ancestry proportion plot of all 229,860 SNPs are shown in Fig 5.

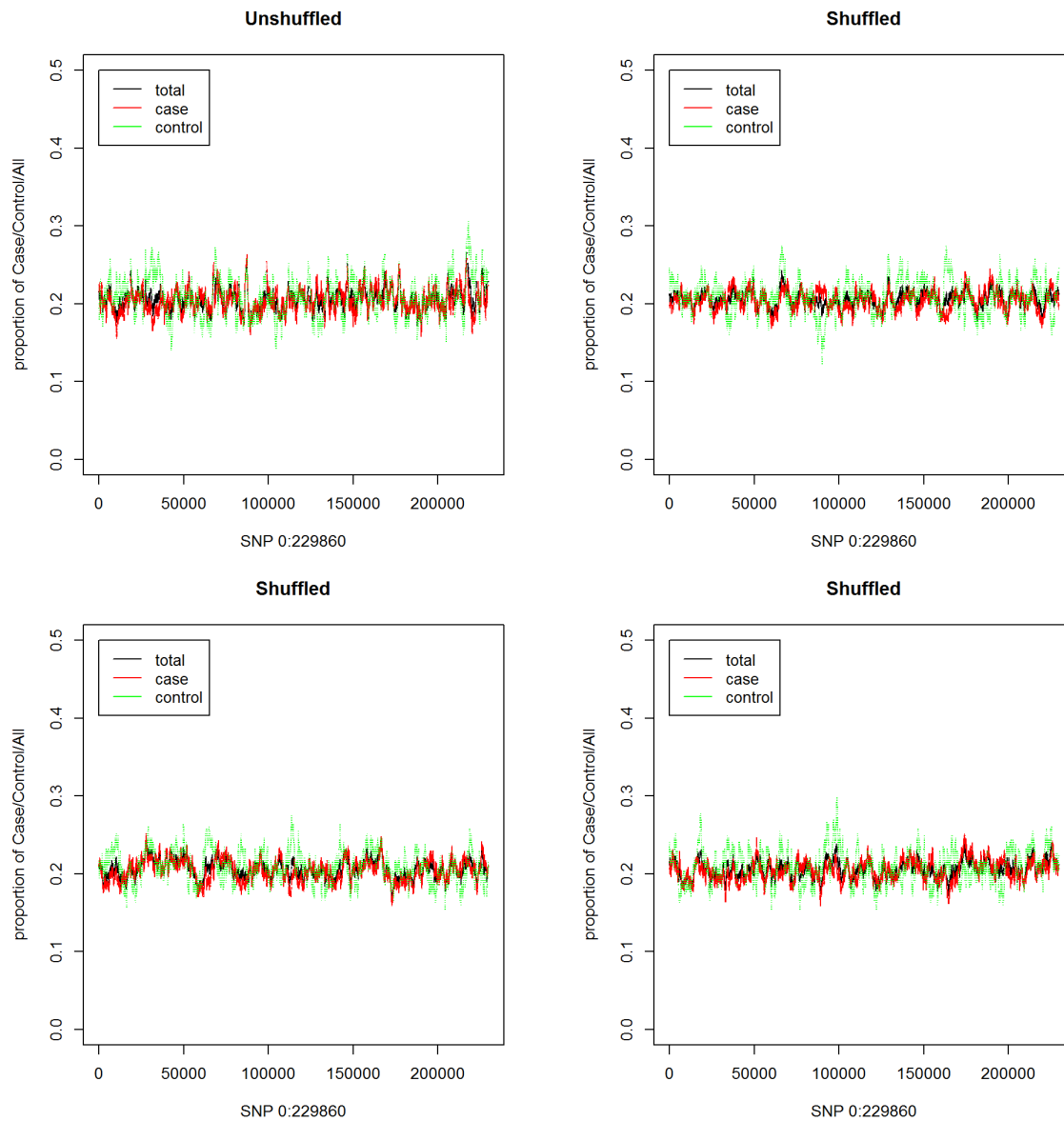


Fig 5 ancestry proportion plot of circle shuffled data.



After circle-shuffling.

Pass the matrix to mixscore for ADM analysis, get the result.

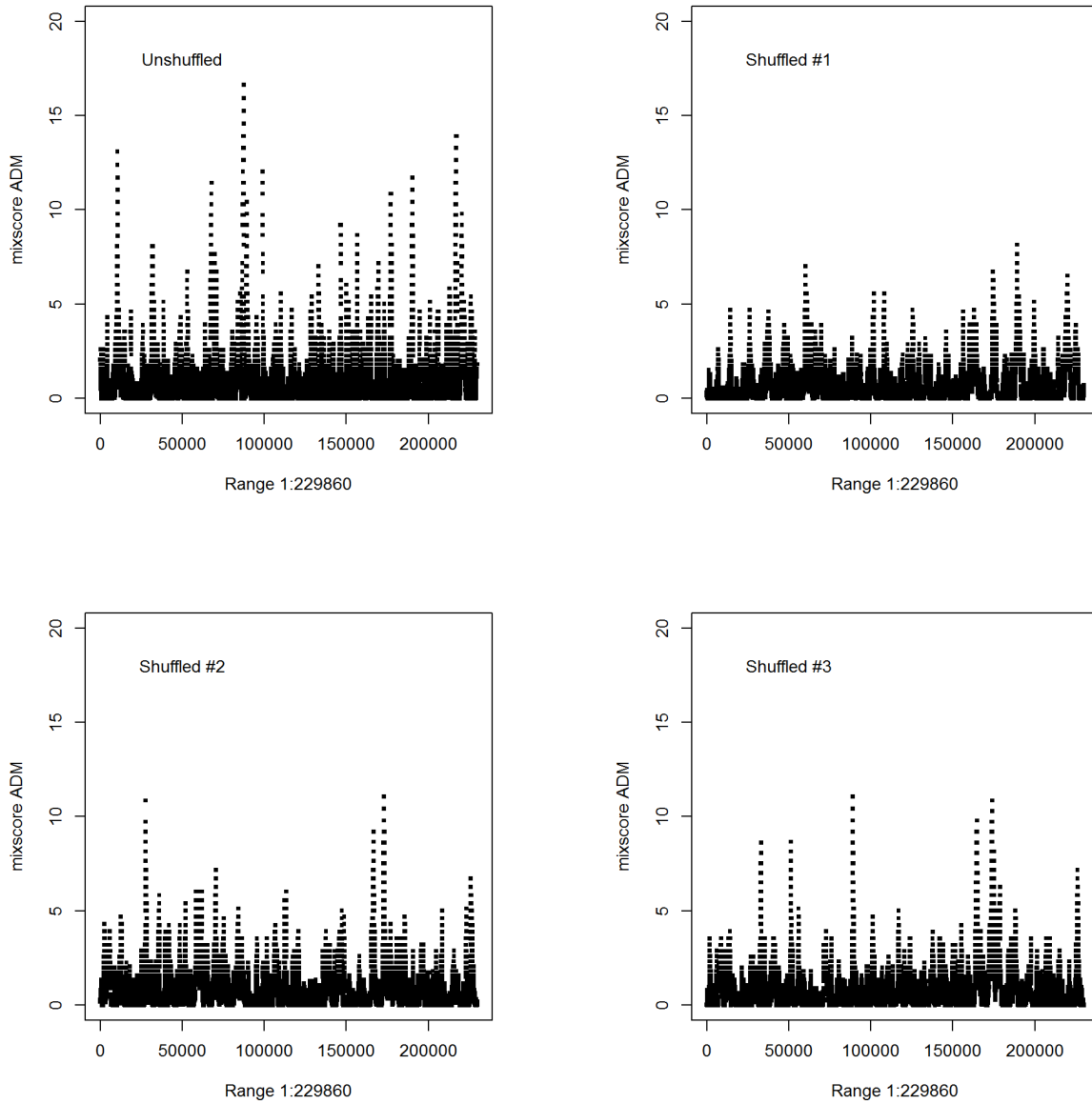


Fig 6 mixscore ADM results of circle-shuffled datasets.

From the results, we could see the mixscore results from circle-shuffled datasets have smaller variance.

|               | Mean           | 20% Trimmed Mean | Median        |
|---------------|----------------|------------------|---------------|
| sampsize.n50  |                |                  |               |
| prop.out.00   | -0.001 (0.019) | 0.003 (0.022)    | 0.007 (0.031) |
| prop.out.05   | 0.124 (0.035)  | 0.051 (0.027)    | 0.048 (0.035) |
| prop.out.10   | 0.322 (0.121)  | 0.154 (0.048)    | 0.130 (0.051) |
| sampsize.n100 |                |                  |               |
| prop.out.00   | -0.001 (0.010) | 0.001 (0.011)    | 0.002 (0.015) |
| prop.out.05   | 0.158 (0.034)  | 0.069 (0.016)    | 0.057 (0.019) |
| prop.out.10   | 0.327 (0.117)  | 0.164 (0.041)    | 0.139 (0.038) |
| sampsize.n150 |                |                  |               |
| prop.out.00   | 0.001 (0.006)  | 0.002 (0.007)    | 0.000 (0.010) |
| prop.out.05   | 0.174 (0.036)  | 0.082 (0.015)    | 0.072 (0.016) |
| prop.out.10   | 0.326 (0.112)  | 0.160 (0.034)    | 0.136 (0.030) |