Assignment 5

109561770

Mohamad Mahdi ziaee

Spring 2016

Contents

[Description 1](#_Toc446340997)

[Application specifications 1](#_Toc446340998)

[Analysis of generated Graphs 2](#_Toc446340999)

[Mappability-output-205 4](#_Toc446341000)

[Source code 0](#_Toc446341001)

[com.bio.main.CheckMappabilityApp 0](#_Toc446341002)

[com.bio.main.GenerateReadsApp 1](#_Toc446341003)

[com.bio.main.MergeOutputApp 2](#_Toc446341004)

[com.bio.main.util.FileUtils 3](#_Toc446341005)

[com.bio.main.util.MappabilityUtils 9](#_Toc446341006)

[com.bio.pojo.Gene 13](#_Toc446341007)

[com.bio.pojo.Mappability 14](#_Toc446341008)

# Description

The objective of this assignment is to generate mappability of 250 gene sequences in scope of HG19.chr1. There are 3 different applications made for this purpose and here is a step by step guidance on how to achieve the goal (Take note that all the files must be placed under Assignment4/io folder as this is the main io folder for both input and output):

1. Generate Reads with different length by running the following application: /Assignment5/src/com/bio/main/GenerateReadsApp.java
2. Use the read files (reads50.fa for example) one by one and generate BowTie outputs and place them in io folder (length 50 BT output is called final-BTout50-v2-m1 in this assignment. Same naming convention was used for the rest).
3. Generate the mappability files by running the following application: /Assignment5/src/com/bio/main/CheckMappabilityApp.java
4. Merge the 3 mappability files (for example: final-output-50) generated from step 3 by running the following application: /Assignment5/src/com/bio/main/MergeOutputApp.java
5. The final file which can be used by R application will be named: Mappability-output-250

## 

## Application specifications

**Programming language:** JAVA

**Other required installations:** JRE 1.8 to run the application,

JDK 1.8 to compile the application.

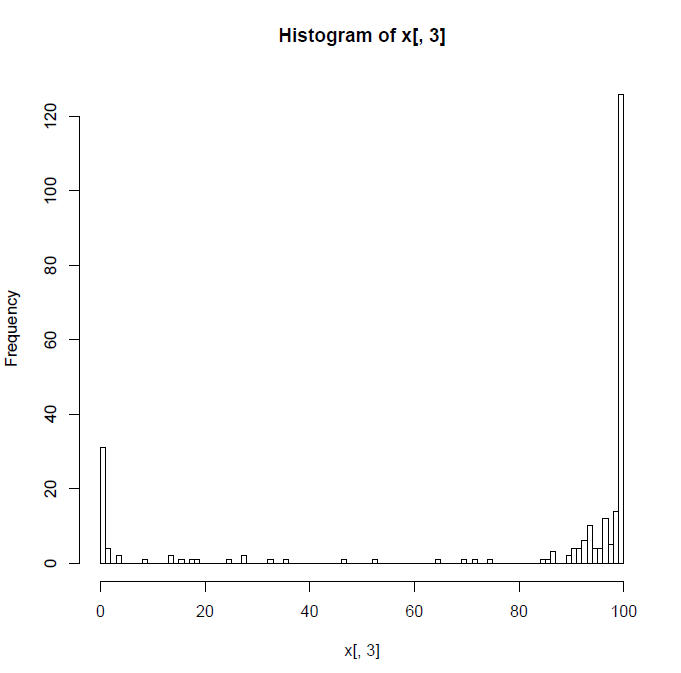
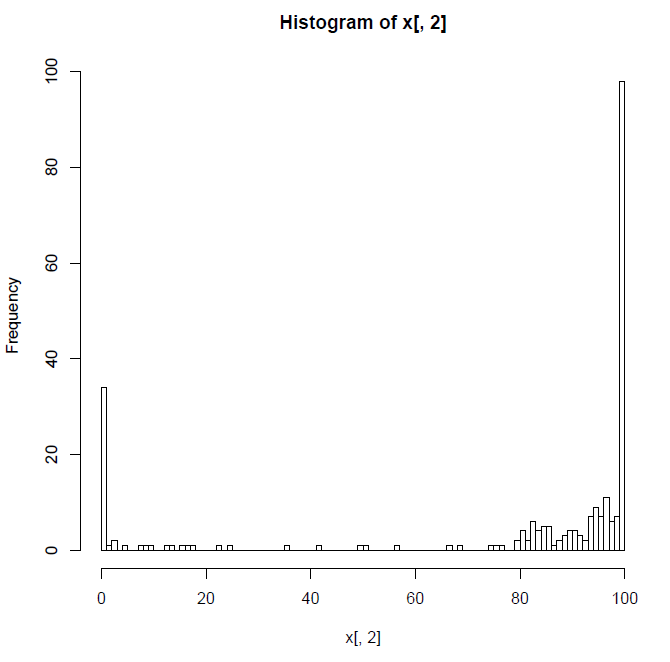
**Version Control:** Git (GitHub) - https://github.com/momazia/Bioinformatic/tree/master/Assignment5

**IDE:** Eclipse (Mars 4.5.1)

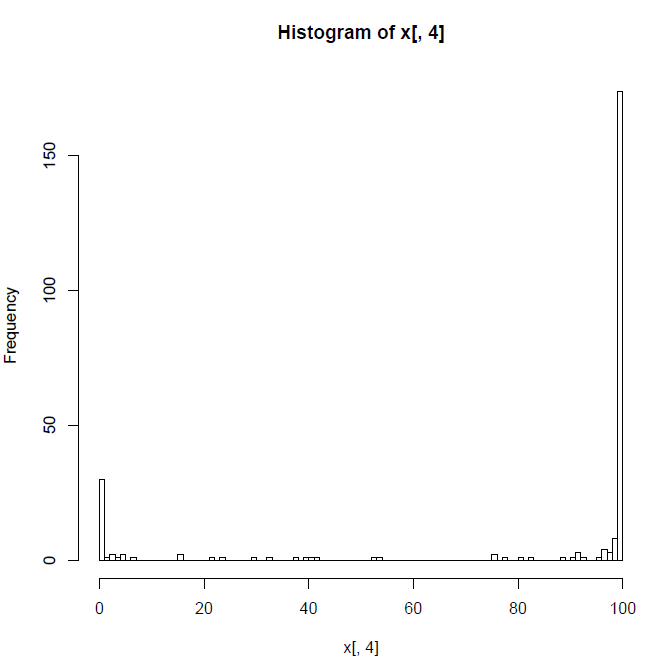
**Build automation tool:** Maven (Refer to /Assignment5/pom.xml for the list of the libraries used)

## Analysis of generated Graphs

As an individual study on each of mappabilities of the different lengths, by looking at their Histogram graphs, the most mappabilities are distributed on either 100 or zero percent. It also shows that 100% mappabilities in all 3 different lengths are having the most occurrences in the output.

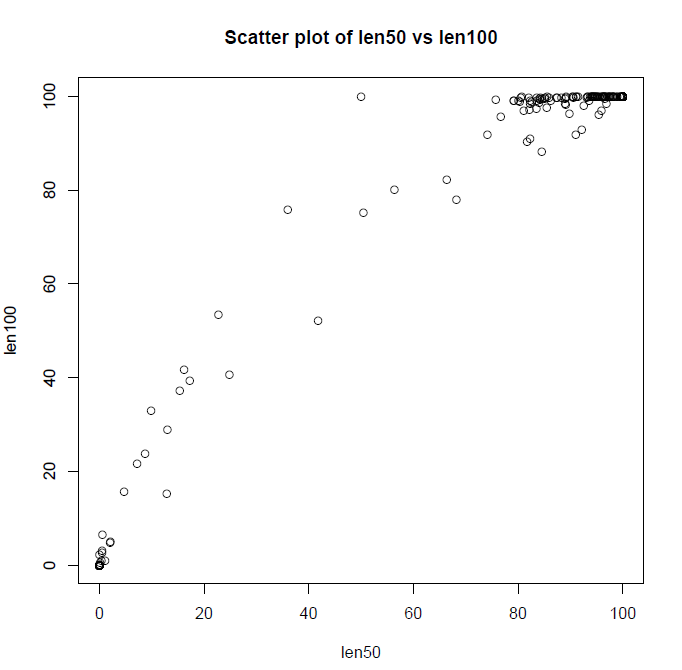
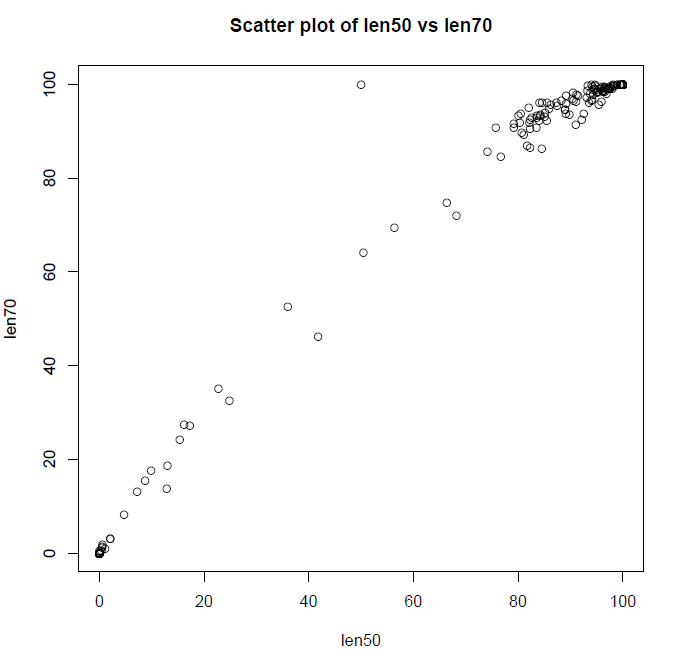


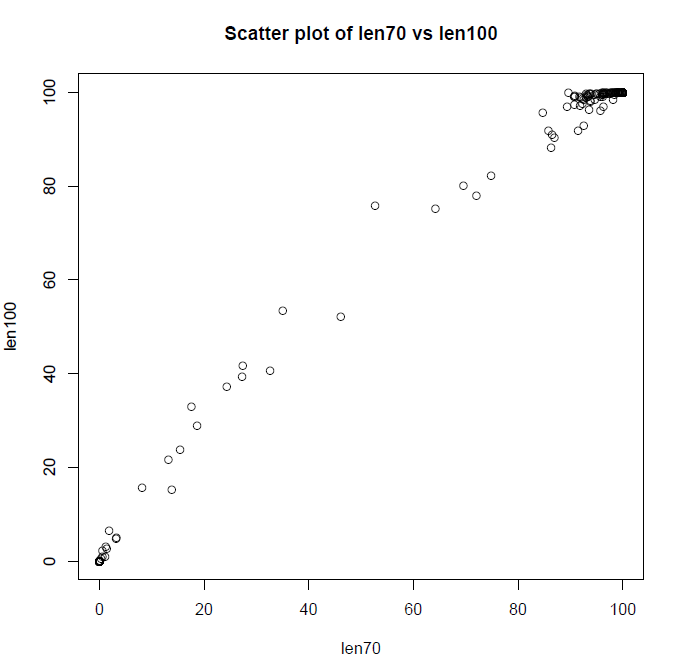
Length 50 Length 70



Length 100

By comparing scatter plot graphs of all the possible combinations between the 3 different read lengths, it is most obvious that the longer the length of a read, the higher the mappability of the read is. Because in none of the graphs below, we have a single dot under the line , where the smaller read length is presented on in all the graphs.





# Mappability-output-205

geneID readLen50 readLen70 readLen100

chr1.19629201.19638640.NM\_003689.- 85.4633 92.2625 97.5375

chr1.32671235.32674288.NM\_001160042.+ 95.0732 98.559 99.8984

chr1.1102483.1102578.NR\_029639.+ 100.00 100.00 100.00

chr1.11979644.11986485.NM\_138346.- 93.0948 97.1057 99.7478

chr1.26364513.26372604.NM\_032513.- 96.1701 98.367 100.00

chr1.21619782.21626362.NR\_038404.+ 96.4324 98.756 100.00

chr1.12916940.12921764.NM\_023014.+ 22.7435 35.0368 53.3757

chr1.28562601.28564616.NM\_178190.+ 100.00 100.00 100.00

chr1.15764937.15773153.NM\_007272.+ 91.1106 97.8274 100.00

chr1.1849028.1850740.NM\_178545.- 100.00 100.00 100.00

chr1.11104854.11107296.NM\_139208.- 94.6929 99.9157 100.00

chr1.32712817.32714461.NM\_032648.+ 100.00 100.00 100.00

chr1.6475293.6479979.NM\_019089.- 93.9616 96.5995 100.00

chr1.12998301.13007406.NM\_001010889.- 0.3644 0.5865 0.9216

chr1.29445936.29450421.NM\_001003682.- 100.00 100.00 100.00

chr1.28975111.28975246.NR\_004407.+ 100.00 100.00 100.00

chr1.16558181.16563659.NM\_030907.- 88.9667 94.6201 99.4237

chr1.10509775.10512060.NM\_001302.+ 100.00 100.00 100.00

chr1.22979681.22988029.NM\_000491.+ 98.2649 99.7826 100.00

chr1.20140521.20141771.NM\_019062.- 100.00 100.00 100.00

chr1.32827861.32829924.NM\_052841.+ 100.00 100.00 100.00

chr1.323891.328581.NR\_028327.+ 0.00 0.541 2.3524

chr1.19609056.19615280.NM\_012067.- 50.413 64.1755 75.151

chr1.1103242.1103332.NR\_029834.+ 100.00 100.00 100.00

chr1.69090.70008.NM\_001005484.+ 100.00 100.00 100.00

chr1.33360195.33366953.NM\_033504.- 99.7019 100.00 100.00

chr1.23046009.23046091.NR\_039832.+ 100.00 100.00 100.00

chr1.6304251.6305638.NM\_001024598.+ 100.00 100.00 100.00

chr1.11917520.11918992.NM\_002521.- 100.00 100.00 100.00

chr1.13421175.13428190.NM\_001098376.+ 0.00 0.00 0.00

chr1.3044538.3044599.NR\_036215.+ 100.00 100.00 100.00

chr1.2460183.2461684.NM\_001010926.- 100.00 100.00 100.00

chr1.28905254.28905334.NR\_003077.- 100.00 100.00 100.00

chr1.13668268.13673511.NM\_001099854.- 0.00 0.00 0.00

chr1.13521962.13526943.NM\_001100114.+ 0.5272 1.364 2.6833

chr1.27695600.27701315.NM\_173452.- 83.551 92.9685 99.7685

chr1.910578.917473.NR\_027693.- 81.6681 86.9323 90.362

chr1.6524170.6526255.NM\_001039664.- 100.00 100.00 100.00

chr1.1072396.1079434.NR\_038869.+ 68.1786 71.9902 77.9939

chr1.28906892.28907024.NR\_002976.- 93.9759 100.00 100.00

chr1.2938045.2939467.NM\_080431.+ 100.00 100.00 100.00

chr1.26798901.26803133.NM\_005517.+ 82.2854 86.4761 91.0719

chr1.1309109.1310818.NM\_017900.- 100.00 100.00 100.00

chr1.20246799.20250110.NM\_014589.- 100.00 100.00 100.00

chr1.12251761.12251839.NR\_039775.+ 100.00 100.00 100.00

chr1.20301923.20306152.NM\_001161729.- 100.00 100.00 100.00

chr1.23695710.23698330.NR\_033691.+ 100.00 100.00 100.00

chr1.16375283.16383803.NM\_001165945.+ 66.3322 74.8077 82.1636

chr1.23337326.23342343.NM\_001242521.- 80.475 93.6742 99.634

chr1.1631377.1633247.NR\_002946.+ 2.0868 3.2204 4.9689

chr1.5624130.5624203.NR\_039612.+ 100.00 100.00 100.00

chr1.32224260.32224336.NR\_036216.- 100.00 100.00 100.00

chr1.26644410.26647014.NM\_001803.+ 80.7045 89.6252 100.00

chr1.12907260.12908578.NM\_001013631.- 17.2577 27.2218 39.4586

chr1.1104384.1104467.NR\_029957.+ 100.00 100.00 100.00

chr1.17007749.17007823.NR\_037446.+ 0.00 0.00 0.00

chr1.32666201.32670991.NM\_024296.+ 94.1995 99.4705 100.00

chr1.13716087.13719064.NM\_001099851.+ 0.0683 0.0688 0.0695

chr1.1470157.1475740.NM\_001114748.- 99.6025 100.00 100.00

chr1.13359818.13369057.NM\_001013407.+ 0.00 0.00 0.00

chr1.934343.935552.NM\_001142467.- 100.00 100.00 100.00

chr1.19578074.19586622.NM\_016183.+ 93.2463 98.6909 100.00

chr1.13328195.13331692.NM\_001013692.- 0.00 0.00 0.1177

chr1.661138.665731.NR\_028327.- 0.00 0.00 0.00

chr1.16340522.16345285.NM\_014424.- 100.00 100.00 100.00

chr1.3541555.3546694.NM\_182752.+ 93.7328 98.0868 100.00

chr1.11714913.11723384.NM\_033182.+ 82.5813 92.9898 98.9728

chr1.13328832.13331671.NM\_001100631.- 0.00 0.00 0.00

chr1.1243993.1247057.NM\_153339.+ 97.5456 99.0651 100.00

chr1.19619740.19622230.NR\_037847.- 94.5104 99.05 100.00

chr1.32083300.32092919.NM\_001525.+ 98.0146 99.1623 100.00

chr1.28919006.28921088.NM\_001193532.+ 81.0625 89.3691 96.9239

chr1.7907671.7913104.NM\_006786.- 87.2771 96.1409 99.775

chr1.16875408.16875482.NR\_037446.+ 0.00 0.00 0.00

chr1.27992571.27998724.NM\_002038.- 89.1219 95.973 99.8513

chr1.24286300.24289949.NM\_017761.+ 41.75 46.1173 52.2535

chr1.1167628.1170420.NM\_080605.+ 100.00 100.00 100.00

chr1.2518188.2522908.NM\_001195738.+ 100.00 100.00 100.00

chr1.23884420.23886285.NM\_002167.- 100.00 100.00 100.00

chr1.17300998.17308081.NM\_001135247.- 92.1524 92.5292 92.9267

chr1.13447413.13452656.NM\_001024661.- 0.00 0.00 0.00

chr1.16068916.16074477.NM\_001013641.+ 85.5406 96.1945 100.00

chr1.26146444.26150097.NR\_024498.- 92.5361 93.7779 98.0304

chr1.31460911.31461481.NR\_002762.+ 100.00 100.00 100.00

chr1.13421175.13428191.NM\_001010890.+ 0.00 0.00 0.00

chr1.16133656.16134194.NM\_001089591.- 35.9918 52.6652 75.8542

chr1.24122088.24127034.NM\_000403.- 96.4672 99.2208 100.00

chr1.621095.622034.NM\_001005221.- 0.00 0.00 0.00

chr1.12884467.12891264.NM\_001146344.- 9.84 17.5832 32.8904

chr1.1361507.1363167.NM\_001146685.+ 98.8827 100.00 100.00

chr1.17185443.17185516.NR\_037446.- 50.00 100.00 100.00

chr1.26485510.26489119.NM\_024869.+ 94.6348 99.096 100.00

chr1.23189651.23189719.NR\_036214.- 100.00 100.00 100.00

chr1.12952726.12958094.NM\_001039361.- 0.00 0.00 0.00

chr1.1353799.1356824.NM\_001145210.- 91.0618 96.2449 100.00

chr1.31971838.31974167.NR\_034112.+ 100.00 100.00 100.00

chr1.934341.935552.NM\_021170.- 100.00 100.00 100.00

chr1.12939032.12946025.NM\_001009611.- 4.7235 8.1745 15.6948

chr1.11114648.11120091.NM\_003132.- 86.244 95.6829 99.1205

chr1.33797993.33798093.NR\_037400.- 100.00 100.00 100.00

chr1.32674694.32681797.NM\_001099434.+ 79.1466 91.7117 99.1005

chr1.12977512.12980233.NM\_001012277.+ 0.00 0.00 0.00

chr1.12834983.12838048.NM\_001080830.+ 82.1286 91.8224 97.2353

chr1.26226606.26232993.NM\_005563.- 97.2073 99.0662 99.9364

chr1.24104875.24114722.NM\_020362.+ 82.2617 92.5343 98.5638

chr1.19209687.19209774.NR\_039844.- 100.00 100.00 100.00

chr1.28832454.28837404.NR\_036473.+ 90.4917 98.1766 100.00

chr1.24018268.24022915.NM\_000975.+ 98.8908 100.00 100.00

chr1.28286503.28294604.NM\_018053.+ 96.5102 99.4771 100.00

chr1.1115076.1121243.NM\_153254.+ 95.3907 95.7035 96.1767

chr1.20808883.20812728.NM\_018584.- 99.9473 100.00 100.00

chr1.31421964.31422052.NR\_004054.- 0.00 0.00 0.00

chr1.2336240.2344010.NM\_153818.- 96.7103 99.13 100.00

chr1.1533387.1535476.NM\_001242659.- 100.00 100.00 100.00

chr1.33352097.33360247.NM\_002143.+ 100.00 100.00 100.00

chr1.33231234.33240571.NM\_001198973.+ 96.5224 98.3815 99.567

chr1.13359832.13368936.NM\_001010889.+ 0.00 0.00 0.00

chr1.24122088.24126060.NM\_001127621.- 95.5901 99.0264 100.00

chr1.24122088.24127294.NM\_001008216.- 96.6453 99.2603 100.00

chr1.27216978.27226962.NM\_022078.- 84.1872 93.2426 99.3829

chr1.28907431.28907565.NR\_003035.- 100.00 100.00 100.00

chr1.31984035.31989846.NR\_027085.+ 82.0028 94.9495 99.8074

chr1.13386646.13390765.NM\_001012276.- 0.00 0.00 0.00

chr1.20301923.20306932.NM\_001161728.- 100.00 100.00 100.00

chr1.852952.854817.NR\_026874.- 100.00 100.00 100.00

chr1.17081400.17090975.NR\_002729.- 7.1699 13.1496 21.6758

chr1.13474052.13477569.NM\_001099850.- 0.00 0.00 0.00

chr1.11708417.11714888.NM\_012168.- 98.038 99.4533 100.00

chr1.28906275.28906405.NR\_002987.- 100.00 100.00 100.00

chr1.27320194.27327377.NM\_001013642.+ 94.2248 98.9598 99.9294

chr1.28918711.28921088.NM\_152304.+ 83.4622 90.7279 97.3222

chr1.24822822.24828850.NR\_045409.- 88.9446 94.63 98.364

chr1.34610.36081.NR\_026818.- 94.6554 99.786 100.00

chr1.27650364.27653016.NR\_033926.+ 82.2897 90.6698 99.0599

chr1.13694888.13698405.NM\_001099850.- 0.00 0.00 0.00

chr1.16062808.16067884.NM\_207348.+ 89.1585 97.6034 99.6986

chr1.17300998.17307173.NM\_001135248.- 90.9892 91.4183 91.8697

chr1.16972068.16976915.NR\_027504.+ 8.6911 15.4039 23.8627

chr1.20825940.20834674.NM\_024544.- 91.4335 97.6111 100.00

chr1.9211726.9211836.NR\_029610.- 100.00 100.00 100.00

chr1.2281852.2284100.NR\_024489.- 100.00 100.00 100.00

chr1.18807423.18812480.NM\_152375.+ 98.4026 99.6391 100.00

chr1.2976180.2984289.NR\_015440.- 100.00 100.00 100.00

chr1.6684924.6693642.NM\_001195753.+ 84.5542 96.1845 99.5591

chr1.1288070.1293915.NM\_032348.- 97.4465 99.2209 100.00

chr1.13108513.13117751.NM\_001013407.- 0.1088 0.3272 0.6565

chr1.22963117.22966175.NM\_015991.+ 100.00 100.00 100.00

chr1.23695463.23698330.NR\_033690.+ 100.00 100.00 100.00

chr1.26496387.26497364.NM\_015871.+ 100.00 100.00 100.00

chr1.2120987.2126214.NM\_182533.- 95.056 98.1776 99.883

chr1.24255559.24255637.NR\_039615.+ 100.00 100.00 100.00

chr1.13742801.13747803.NM\_001100114.+ 0.5249 1.1758 3.0797

chr1.1327667.1334718.NM\_001039577.- 80.0343 93.3114 99.065

chr1.1370902.1378262.NM\_022834.+ 100.00 100.00 100.00

chr1.803450.812182.NR\_027055.- 74.0988 85.7555 91.8916

chr1.28905049.28908366.NR\_024127.- 93.2987 99.7537 100.00

chr1.1309109.1310580.NM\_001127229.- 100.00 100.00 100.00

chr1.11249397.11256038.NM\_021146.+ 96.3441 98.7219 100.00

chr1.1568158.1570027.NR\_002946.+ 2.033 3.1667 4.9153

chr1.2980635.2984289.NR\_024371.- 100.00 100.00 100.00

chr1.1309110.1310562.NM\_001127230.- 100.00 100.00 100.00

chr1.2487207.2488450.NR\_037844.- 100.00 100.00 100.00

chr1.13641972.13648988.NM\_001010890.+ 0.00 0.00 0.00

chr1.19592475.19600568.NR\_040288.- 56.3526 69.5414 80.0976

chr1.32681797.32687926.NM\_019118.- 85.8717 94.9175 99.7015

chr1.1260142.1264276.NM\_001029885.+ 100.00 100.00 100.00

chr1.6489893.6489956.NR\_036218.- 0.00 0.00 0.00

chr1.1007125.1009687.NM\_001205252.- 100.00 100.00 100.00

chr1.12907235.12908237.NM\_001146181.- 13.0115 18.6495 29.0144

chr1.20438433.20446008.NM\_012400.- 95.9208 99.0674 100.00

chr1.24117645.24122029.NM\_007260.+ 100.00 100.00 100.00

chr1.22352041.22357715.NR\_023919.+ 84.2311 93.595 99.7489

chr1.12567299.12567451.NR\_003025.+ 100.00 100.00 100.00

chr1.23685940.23694879.NM\_030634.- 90.5512 96.5614 99.7059

chr1.1266725.1269844.NM\_152228.+ 100.00 100.00 100.00

chr1.1337275.1342693.NM\_017971.- 80.2943 91.905 98.9284

chr1.11873.14408.NR\_046018.+ 100.00 100.00 100.00

chr1.1177825.1182102.NM\_001014980.- 98.1079 100.00 100.00

chr1.11900375.11907673.NR\_037806.+ 89.0606 93.8027 98.1942

chr1.27719151.27722317.NM\_005281.+ 100.00 100.00 100.00

chr1.28655512.28662478.NM\_017638.+ 79.1528 90.7931 99.0826

chr1.11734536.11741271.NM\_006341.- 85.0583 93.1593 99.5931

chr1.32687970.32697205.NM\_003757.+ 75.6913 90.8139 99.3761

chr1.22351706.22357715.NR\_023918.+ 85.1174 93.9562 99.7631

chr1.9712667.9714644.NR\_027045.- 100.00 100.00 100.00

chr1.34326075.34330392.NM\_145205.+ 99.4142 100.00 100.00

chr1.3477259.3477354.NR\_030277.- 100.00 100.00 100.00

chr1.23518387.23521222.NM\_000864.- 94.9031 98.9877 100.00

chr1.6640062.6649340.NM\_005341.+ 90.3456 96.9921 99.9564

chr1.3689351.3692546.NM\_001163724.+ 100.00 100.00 100.00

chr1.26232852.26232945.NR\_037481.- 100.00 100.00 100.00

chr1.27331510.27339333.NM\_052943.- 96.1796 98.6717 100.00

chr1.11905766.11907840.NM\_006172.- 76.642 84.6883 95.6456

chr1.32043745.32053287.NM\_001204415.+ 89.7819 93.5712 96.2512

chr1.1334909.1337426.NR\_015434.+ 100.00 100.00 100.00

chr1.8938893.8939943.NR\_038351.+ 100.00 100.00 100.00

chr1.20878931.20881513.NM\_207334.+ 100.00 100.00 100.00

chr1.1146705.1149548.NM\_003327.- 100.00 100.00 100.00

chr1.27189632.27190947.NM\_006142.+ 100.00 100.00 100.00

chr1.1846265.1848733.NM\_138705.+ 100.00 100.00 100.00

chr1.23370797.23370865.NR\_036057.+ 100.00 100.00 100.00

chr1.26187974.26197744.NM\_178422.- 83.9831 92.2482 98.6144

chr1.27705595.27709805.NM\_207397.- 100.00 100.00 100.00

chr1.29445936.29449013.NM\_001171868.- 100.00 100.00 100.00

chr1.2481358.2484284.NR\_026927.- 24.8523 32.5866 40.7145

chr1.6521213.6526255.NM\_003790.- 94.1318 96.7625 100.00

chr1.1567559.1570030.NM\_006983.+ 12.8819 13.8218 15.2614

chr1.7907671.7913551.NM\_021995.- 88.2524 96.4378 99.7924

chr1.13182959.13184326.NM\_001136561.- 16.1608 27.3498 41.7981

chr1.13495253.13498259.NM\_001045480.+ 1.0484 1.0555 1.0664

chr1.6685209.6693642.NM\_001195752.+ 84.0291 96.0545 99.544

chr1.28421445.28423069.NR\_027268.- 100.00 100.00 100.00

chr1.16330730.16333184.NM\_178840.+ 87.4012 95.5136 99.7877

chr1.28160911.28161077.NR\_002997.+ 100.00 100.00 100.00

chr1.5922731.5922801.NR\_039838.- 100.00 100.00 100.00

chr1.11714431.11723384.NM\_183412.+ 83.5243 93.3701 99.0287

chr1.13607430.13611550.NM\_001012276.- 0.00 0.00 0.00

chr1.31212002.31212079.NR\_039616.- 100.00 100.00 100.00

chr1.13607766.13610487.NM\_001012277.- 0.00 0.00 0.00

chr1.901876.910484.NM\_001160184.+ 96.8688 98.1145 98.4722

chr1.948846.949919.NM\_005101.+ 100.00 100.00 100.00

chr1.31191618.31199593.NR\_034182.+ 100.00 100.00 100.00

chr1.20512577.20519942.NM\_152376.+ 99.6446 100.00 100.00

chr1.12976449.12980568.NM\_001012276.+ 0.00 0.00 0.00

chr1.26881032.26881084.NR\_031740.+ 100.00 100.00 100.00

chr1.15986363.15988217.NM\_006511.+ 100.00 100.00 100.00

chr1.1682670.1690081.NM\_001198995.- 84.5151 86.2844 88.1565

chr1.895966.901099.NM\_198317.+ 95.9284 96.3073 96.8812

chr1.2487804.2495267.NM\_003820.+ 99.5414 100.00 100.00

chr1.26606212.26608013.NM\_031286.+ 100.00 100.00 100.00

chr1.134772.140566.NR\_039983.- 0.5744 1.869 6.4618

chr1.22970117.22974603.NM\_172369.+ 100.00 100.00 100.00

chr1.31408535.31408623.NR\_033295.- 0.00 0.00 0.00

chr1.26226606.26233368.NM\_203399.- 97.3633 99.1185 99.94

chr1.33327868.33336414.NM\_001171940.- 93.5624 96.1779 99.195

chr1.13035542.13038381.NM\_001100631.+ 0.00 0.00 0.00

chr1.26226606.26232644.NM\_203401.- 97.0446 99.0116 99.9326

chr1.367658.368597.NM\_001005221.+ 0.00 0.00 0.00

chr1.761585.762902.NR\_024321.- 100.00 100.00 100.00

chr1.2517898.2522908.NR\_036638.+ 100.00 100.00 100.00

chr1.31441009.31441084.NR\_003066.- 100.00 100.00 100.00

chr1.1138887.1142089.NM\_004195.- 100.00 100.00 100.00

chr1.13641972.13648987.NM\_001098376.+ 0.00 0.00 0.00

chr1.12851545.12856777.NM\_023013.+ 15.3579 24.3269 37.1907

chr1.32826870.32827844.NM\_001167676.- 100.00 100.00 100.00

chr1.32799439.32801834.NM\_023009.- 100.00 100.00 100.00

chr1.20978259.20988037.NM\_005216.- 94.2954 97.7444 100.00

chr1.31838099.31845923.NM\_004102.- 97.7235 99.7679 100.00

chr1.27237974.27240567.NM\_021969.- 95.283 98.5341 100.00

chr1.25568739.25573985.NM\_020317.- 96.0939 99.5171 100.00

# Source code

## com.bio.main.CheckMappabilityApp

package com.bio.main;

import java.io.IOException;

import java.util.List;

import com.bio.main.util.FileUtils;

import com.bio.main.util.MappabilityUtils;

/\*\*

\* The main application to create different mappability files for the given BowTie outputs. All the files must be placed under /Assignment5/io folder.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class CheckMappabilityApp {

// Constants to the file names

private static final String CHR1\_250\_FA = "chr1-250.fa";

private static final int TILE\_LENGTH\_50 = 50;

public static final String FINAL\_OUTPUT\_50 = "final-output-50";

private static final String FINAL\_B\_TOUT50\_V2\_M1 = "final-BTout50-v2-m1";

private static final int TILE\_LENGTH\_70 = 70;

public static final String FINAL\_OUTPUT\_70 = "final-output-70";

private static final String FINAL\_B\_TOUT70\_V2\_M1 = "final-BTout70-v2-m1";

private static final int TILE\_LENGTH\_100 = 100;

public static final String FINAL\_OUTPUT\_100 = "final-output-100";

private static final String FINAL\_B\_TOUT100\_V2\_M1 = "final-BTout100-v2-m1";

/\*\*

\* The main method to be executed.

\*

\* @param args

\*/

public static void main(String[] args) {

try {

// Reading the 250 chromosomes file.

List<String> chr1FileLines = FileUtils.getInstance().readFile(CHR1\_250\_FA);

MappabilityUtils.getInstance().checkMappability(FINAL\_B\_TOUT50\_V2\_M1, TILE\_LENGTH\_50, FINAL\_OUTPUT\_50, GenerateReadsApp.READS\_50\_FA, chr1FileLines);

MappabilityUtils.getInstance().checkMappability(FINAL\_B\_TOUT70\_V2\_M1, TILE\_LENGTH\_70, FINAL\_OUTPUT\_70, GenerateReadsApp.READS\_70\_FA, chr1FileLines);

MappabilityUtils.getInstance().checkMappability(FINAL\_B\_TOUT100\_V2\_M1, TILE\_LENGTH\_100, FINAL\_OUTPUT\_100, GenerateReadsApp.READS\_100\_FA, chr1FileLines);

} catch (IOException e) {

e.printStackTrace();

}

}

}

## com.bio.main.GenerateReadsApp

package com.bio.main;

import java.io.IOException;

import com.bio.main.util.FileUtils;

/\*\*

\* This application reads a chromose file (in this case, chr1-250.fa) and produces different Reads files for the given size and save them into the

\* destination file. All the files must be placed under /Assignment5/io folder.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class GenerateReadsApp {

// The list of constants

private static final String CHR1\_250\_FA = "chr1-250.fa";

public static final String READS\_50\_FA = "reads50.fa";

public static final String READS\_70\_FA = "reads70.fa";

public static final String READS\_100\_FA = "reads100.fa";

private static final int READ\_LENGTH\_50 = 50;

private static final int READ\_LENGTH\_70 = 70;

private static final int READ\_LENGTH\_100 = 100;

/\*\*

\* Main application method to be executed.

\*

\* @param args

\*/

public static void main(String[] args) {

try {

FileUtils.getInstance().createReads(READ\_LENGTH\_50, READS\_50\_FA, CHR1\_250\_FA);

FileUtils.getInstance().createReads(READ\_LENGTH\_70, READS\_70\_FA, CHR1\_250\_FA);

FileUtils.getInstance().createReads(READ\_LENGTH\_100, READS\_100\_FA, CHR1\_250\_FA);

} catch (IOException e) {

e.printStackTrace();

}

}

}

## com.bio.main.MergeOutputApp

package com.bio.main;

import java.io.IOException;

import com.bio.main.util.FileUtils;

/\*\*

\* Merges the given mappability files into one file which could be used by R application for analysis.All the files must be placed under

\* /Assignment5/io folder.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class MergeOutputApp {

private static final String MAPPABILITY\_OUTPUT\_250 = "Mappability-output-250";

/\*\*

\* The main method to be executed.

\*

\* @param str

\*/

public static void main(String[] str) {

try {

FileUtils.getInstance().merge(MAPPABILITY\_OUTPUT\_250, CheckMappabilityApp.FINAL\_OUTPUT\_50, CheckMappabilityApp.FINAL\_OUTPUT\_70, CheckMappabilityApp.FINAL\_OUTPUT\_100);

} catch (IOException e) {

e.printStackTrace();

}

}

}

## com.bio.main.util.FileUtils

package com.bio.main.util;

import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.nio.file.Files;

import java.nio.file.Paths;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.concurrent.TimeUnit;

import org.apache.commons.lang3.StringUtils;

import org.apache.commons.lang3.time.StopWatch;

import com.bio.pojo.Mappability;

/\*\*

\* The main utility class which is in charge of all the file related operations

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class FileUtils {

// Constants used for reading from and writing to the files.

private static final String ZERO = "0.00";

private static final String GENE\_ID\_HEADER = "geneID";

public static final String SEPARATOR\_DOT = ".";

public static final String SEPARATOR\_TAB = "\t";

public static final String CHR\_INDICATOR = ">";

public static final String IO\_PATH = "../Assignment5/io/";

private static FileUtils instance = null;

/\*\*

\* Private constructor for Singleton design pattern purpose. Declared private so it is not accessible from outside.

\*/

private FileUtils() {

}

/\*\*

\* Gets instance of this class. It will instantiate it if it is not done yet, once.

\*

\* @return

\*/

public static FileUtils getInstance() {

if (instance == null) {

instance = new FileUtils();

}

return instance;

}

/\*\*

\* Generates Reads for the given read length and save it into the output file name passed to this method. If the file output exists, it will

\* overwrite it.

\*

\* @param readLength

\* The read length

\* @param outputFileName

\* The output file name to save the reads to

\* @param chr1FileName

\* The chromosome source file to generate the reads from

\* @throws IOException

\*/

public void createReads(int readLength, String outputFileName, String chr1FileName) throws IOException {

System.out.println("Creating [" + outputFileName + "]...");

StopWatch stopWatch = new StopWatch();

stopWatch.start();

String outputPath = FileUtils.IO\_PATH + outputFileName;

Files.deleteIfExists(Paths.get(outputPath));

List<String> lines = readFile(chr1FileName);

String header = null;

PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(outputPath, true)));

for (int i = 0; i < lines.size(); i++) {

String line = lines.get(i);

if (i % 2 == 0) {

// Reading the gene name

header = line;

} else {

// Reading the sequence

List<String> tiles = createTiles(readLength, line);

writeReadsToFile(header, tiles, out);

}

}

out.close();

stopWatch.stop();

System.out.println("[" + outputFileName + "] is done in [" + TimeUnit.MILLISECONDS.convert(stopWatch.getNanoTime(), TimeUnit.NANOSECONDS) + "] MILLISECONDS");

}

/\*\*

\* Writes the tiles given into the output file by adding a counter at the end of the gene name.

\*

\* @param header

\* Gene name

\* @param tiles

\* The list of tiles generated for a given gene sequence

\* @param out

\* the output file PrintWriter

\* @throws IOException

\*/

private void writeReadsToFile(String header, List<String> tiles, PrintWriter out) throws IOException {

int counter = 0;

for (String tile : tiles) {

// Writing the header

StringBuffer buffer = new StringBuffer();

buffer.append(header);

buffer.append(SEPARATOR\_DOT);

buffer.append(counter++);

out.println(buffer.toString());

out.println(tile);

}

}

/\*\*

\* Given a gene sequence and a read length, it generates all the possible tiles.

\*

\* @param readLength

\* Desired read length

\* @param line

\* The main string to create the tiles from

\* @return The list of tiles

\*/

public List<String> createTiles(int readLength, String line) {

List<String> result = new ArrayList<>();

if (readLength < line.length()) {

for (int index = 0; index < line.length() - readLength + 1; index++) {

result.add(line.substring(index, index + readLength));

}

} else {

result.add(line);

}

return result;

}

/\*\*

\* Given the file name, it reads all its content and put it into a list.

\*

\* @param fileName

\* File name to read

\* @return

\* @throws IOException

\*/

public List<String> readFile(String fileName) throws IOException {

return Files.readAllLines(Paths.get(FileUtils.IO\_PATH + fileName));

}

/\*\*

\* Writes the given map of mappabilities into the output file.

\*

\* @param result

\* A map which the key is the gene names and the value is the mappability related to that gene.

\* @param outputFileName

\* The name of the file to save the result into.

\* @param fileTileLength

\* The length of the tile, this is used to generate the header

\* @throws IOException

\*/

public void writeFile(Map<String, Mappability> result, String outputFileName, int fileTileLength) throws IOException {

String path = FileUtils.IO\_PATH + outputFileName;

Files.deleteIfExists(Paths.get(path));

PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(path, true)));

// Printing the header

out.println(String.join(SEPARATOR\_TAB, GENE\_ID\_HEADER, "readLen" + fileTileLength));

for (String geneName : result.keySet()) {

Mappability output = result.get(geneName);

if (output == null) {

out.println(String.join(SEPARATOR\_TAB, geneName, ZERO));

} else {

out.println(String.join(SEPARATOR\_TAB, geneName, output.getMappability()));

}

}

out.close();

}

/\*\*

\* Merges any given number of files into one. It overwrites if the output already exists.

\*

\* @param outputFileName

\* The output file name.

\* @param inputFileNames

\* The list of input File names

\* @throws IOException

\*/

public void merge(String outputFileName, String... inputFileNames) throws IOException {

Map<String, List<String>> file = readFileIntoMap(inputFileNames);

String path = FileUtils.IO\_PATH + outputFileName;

Files.deleteIfExists(Paths.get(path));

PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(path, true)));

// Printing the header

StringBuffer stringBuffer = createLine(file, GENE\_ID\_HEADER);

out.println(stringBuffer);

file.remove(GENE\_ID\_HEADER);

// Printing the body

for (String geneId : file.keySet()) {

stringBuffer = createLine(file, geneId);

out.println(stringBuffer);

}

out.close();

}

/\*\*

\* It constructs a line in merging process for each mappability type.

\*

\* @param file

\* The content of the whole mappabilities.

\* @param geneId

\* the Desired geneId which the line must be created for.

\* @return

\*/

private StringBuffer createLine(Map<String, List<String>> file, String geneId) {

StringBuffer stringBuffer = new StringBuffer();

stringBuffer.append(geneId);

stringBuffer.append(SEPARATOR\_TAB);

stringBuffer.append(String.join(SEPARATOR\_TAB, file.get(geneId)));

return stringBuffer;

}

/\*\*

\* For the given file names, it reads the files and put them into a map in which the key is the gene names and the value is the mappability

\* related to that gene.

\*

\* @param inputFileNames

\* @return

\* @throws IOException

\*/

private Map<String, List<String>> readFileIntoMap(String[] inputFileNames) throws IOException {

Map<String, List<String>> result = new HashMap<>();

for (String fileName : inputFileNames) {

List<String> lines = readFile(fileName);

for (String line : lines) {

String[] splittedLine = StringUtils.splitByWholeSeparator(line, SEPARATOR\_TAB);

String geneId = splittedLine[0];

if (!result.containsKey(geneId)) {

result.put(geneId, new ArrayList<>());

}

List<String> existingMappabilities = result.get(geneId);

existingMappabilities.add(splittedLine[1]);

result.put(geneId, existingMappabilities);

}

}

return result;

}

}

## com.bio.main.util.MappabilityUtils

package com.bio.main.util;

import java.io.IOException;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import java.util.concurrent.TimeUnit;

import org.apache.commons.lang3.StringUtils;

import org.apache.commons.lang3.time.StopWatch;

import com.bio.pojo.Gene;

import com.bio.pojo.Mappability;

/\*\*

\* The main utility class which is in charge of mappability related operations.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class MappabilityUtils {

private static MappabilityUtils instance = null;

/\*\*

\* Private constructor for Singleton design pattern purpose. Declared private so it is not accessible from outside.

\*/

private MappabilityUtils() {

}

/\*\*

\* Gets instance of this class. It will instantiate it if it is not done yet, once.

\*

\* @return

\*/

public static MappabilityUtils getInstance() {

if (instance == null) {

instance = new MappabilityUtils();

}

return instance;

}

/\*\*

\* The main method to check the mappabilities of the given read file names.

\*

\* @param btOutputFileName

\* BowTie output file name

\* @param fileTileLength

\* Tile length

\* @param outputFileName

\* Output file name which the result will be saved into

\* @param readFileName

\* Reads file name

\* @param chr1FileLines

\* chromosome file name

\* @throws IOException

\*/

public void checkMappability(String btOutputFileName, int fileTileLength, String outputFileName, String readFileName, List<String> chr1FileLines) throws IOException {

System.out.println("Creating [" + outputFileName + "]...");

StopWatch stopWatch = new StopWatch();

stopWatch.start();

System.out.println("Reading [" + btOutputFileName + "]");

List<String> btOutputFileLines = FileUtils.getInstance().readFile(btOutputFileName);

System.out.println("Done reading [" + btOutputFileName + "]");

System.out.println("Reading [" + readFileName + "]");

List<String> readFileLines = FileUtils.getInstance().readFile(readFileName);

System.out.println("Done reading [" + readFileName + "]");

Map<String, Mappability> result = createMappability(btOutputFileLines, readFileLines, chr1FileLines);

FileUtils.getInstance().writeFile(result, outputFileName, fileTileLength);

stopWatch.stop();

System.out.println("[" + outputFileName + "] is done in [" + TimeUnit.MILLISECONDS.convert(stopWatch.getNanoTime(), TimeUnit.NANOSECONDS) + "] MILLISECONDS");

}

/\*\*

\* Creating a map which its key is a gene name and its value is the mappability. It also calculates the number of tiles which are within the gene.

\*

\* @param btOutputFileLines

\* BowTie output file name

\* @param readFileLines

\* Reads file content in a list

\* @param chr1FileLines

\* chromosome file name in a list

\* @return

\* @throws IOException

\*/

private Map<String, Mappability> createMappability(List<String> btOutputFileLines, List<String> readFileLines, List<String> chr1FileLines) throws IOException {

Map<String, Mappability> result = new HashMap<>();

for (String btOutputLine : btOutputFileLines) {

String[] splittedLine = StringUtils.split(btOutputLine, FileUtils.SEPARATOR\_TAB);

Integer tileStart = getTileStartIndex(splittedLine);

Integer tileEnd = getTileEndIndex(splittedLine);

Gene gene = getGeneId(splittedLine);

String geneName = gene.getName();

if (!result.containsKey(geneName)) {

result.put(geneName, new Mappability(0, getTotalNumberOfTiles(geneName, readFileLines)));

}

Mappability output = result.get(geneName);

if (isTileWithinGene(tileStart, tileEnd, gene)) {

output.addNumberOfMatches();

}

result.put(geneName, output);

}

// Add the missing tiles from output

for (int lineIndex = 0; lineIndex < chr1FileLines.size(); lineIndex++) {

if (lineIndex % 2 == 0) {

// Reading the first line

String geneName = StringUtils.substringAfter(chr1FileLines.get(lineIndex), FileUtils.CHR\_INDICATOR);

if (!result.containsKey(geneName)) {

result.put(geneName, null);

}

}

chr1FileLines.get(lineIndex);

}

return result;

}

/\*\*

\* Calculates the total number of tiles by counting the occurance of the gene name in the reads file content list passed.

\*

\* @param geneName

\* @param readFileLines

\* @return

\* @throws IOException

\*/

private Integer getTotalNumberOfTiles(String geneName, List<String> readFileLines) throws IOException {

int counter = 0;

for (String line : readFileLines) {

if (line.contains(geneName)) {

counter++;

}

}

return counter;

}

/\*\*

\* A tile is within a gene if its start and end indexes are within the gene's index range.

\*

\* @param tileStart

\* @param tileEnd

\* @param gene

\* @return

\*/

private boolean isTileWithinGene(Integer tileStart, Integer tileEnd, Gene gene) {

return gene.getStartIndex() <= tileStart && tileEnd <= gene.getEndIndex();

}

/\*\*

\* Extracts the geneID out of the splitted file line.

\*

\* @param splittedLine

\* @return

\*/

private Gene getGeneId(String[] splittedLine) {

String[] geneSplittedStr = StringUtils.split(splittedLine[3], FileUtils.SEPARATOR\_DOT);

return new Gene(Integer.valueOf(geneSplittedStr[1]), Integer.valueOf(geneSplittedStr[2]), StringUtils.substringBeforeLast(splittedLine[3], FileUtils.SEPARATOR\_DOT));

}

/\*\*

\* Gets the tile end index by looking at the 3rd element in the given splitted line

\*

\* @param splittedLine

\* @return

\*/

private Integer getTileEndIndex(String[] splittedLine) {

return Integer.valueOf(splittedLine[2]);

}

/\*\*

\* Gets the tile end index by looking at the 2nd element in the given splitted line

\*

\* @param splittedLine

\* @return

\*/

private Integer getTileStartIndex(String[] splittedLine) {

return Integer.valueOf(splittedLine[1]);

}

}

## com.bio.pojo.Gene

package com.bio.pojo;

/\*\*

\* A POJO to hold gene information like start and end indexes.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class Gene {

private Integer startIndex;

private Integer endIndex;

private String name;

/\*\*

\* The main constructor.

\*

\* @param startIndex

\* @param endIndex

\* @param name

\*/

public Gene(Integer startIndex, Integer endIndex, String name) {

this.startIndex = startIndex;

this.endIndex = endIndex;

this.name = name;

}

public Integer getStartIndex() {

return startIndex;

}

public void setStartIndex(Integer startIndex) {

this.startIndex = startIndex;

}

public Integer getEndIndex() {

return endIndex;

}

public void setEndIndex(Integer endIndex) {

this.endIndex = endIndex;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

## com.bio.pojo.Mappability

package com.bio.pojo;

import java.text.DecimalFormat;

/\*\*

\* A POJO to hold the data related to the mappability. It contains the number of matches and the tiles.

\*

\* @author Mohamad Mahdi Ziaee

\*

\*/

public class Mappability {

private Integer numberOfMatches;

private Integer numberOfTiles;

private static final DecimalFormat df = new DecimalFormat("#0.00##");

/\*\*

\* The main constructor.

\*

\* @param numberOfMatches

\* @param numberOfTiles

\*/

public Mappability(Integer numberOfMatches, Integer numberOfTiles) {

this.numberOfMatches = numberOfMatches;

this.numberOfTiles = numberOfTiles;

}

public void addNumberOfMatches() {

this.numberOfMatches++;

}

public Integer getNumberOfTiles() {

return numberOfTiles;

}

public void setNumberOfTiles(Integer numberOfTiles) {

this.numberOfTiles = numberOfTiles;

}

public Integer getNumberOfMatches() {

return numberOfMatches;

}

public void setNumberOfMatches(Integer numberOfMatches) {

this.numberOfMatches = numberOfMatches;

}

/\*\*

\* Calculates the mappability percentage by dividing numberOfMatches over numberOfTiles \* 100. It is formatted using #0.00## formatter.

\*

\* @return

\*/

public String getMappability() {

return df.format(numberOfMatches.doubleValue() / numberOfTiles.doubleValue() \* 100);

}

}