**Bio Informatics Computing**

MUKHASIR SHAH SYED

109137268

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Date : May - 04 - 2015 Author : Mukhasir Shah Syed

\* RNA-Seq Data analysis pipeline should include atleast the following three process:

\* 1. exon region mask complement.

\* 2. junction region preparation.

\* 3. mapping on exon and junction regions, and gene-level read counting.

\* In this assignment, we will work on step 1 and 3 by processing without considering junction region mappings.

\* Task 1: exon region mask complement

\* For this we will used only HG19 chr1

\* 1. Use HG19 RefSeq exon annotation file and trim it for chr1 only with 6-columns each.

\* This would be done using awk commands in Unix.

\* 2. From resulting annotation from above step, collapse all overlapped regions and make a collapsed annotation with each collapsed exon name "X" and strand "+".

\* For Example:

\* chr1 6469 6628 NR\_024540\_exon\_3\_0\_chr1\_6470\_r 0 -

chr1 6469 6631 NR\_028269\_exon\_3\_0\_chr1\_6470\_r 0 -

becomes

chr1 6469 6631 x 0 +

Code will be written to collapse all overlapped regions and write the annotation into a file.

\* 3. By using collapsed exon annotation, mask all non-exon regions of HG19 chr1 with 'N's.

Code will be written to mask all non-exon regions of chr1 with 'N' and write file value into file.

\* Task 2: read mapping and read count.

\* 1. Using Bowtie, map a reads file (fastq format) onto th genome. Collapsed exon annotation file will would be used to create database/build and on that build we will map reads that are provided in a file(ERR030893-1.fq) by instructor.

\* 2. Using the original exon annotation file, count mapped reads on each exon.

\* Code will be written to check if map is between exon length then increment count by 1 and make list.

\* 3. Convert exon-level read counting to gene-level counting i.e, gene expression level.

\* Code will be written to make list of genes from exons, which contains GeneID and reads\_count.

\* To complie/run the code we have to create a Console Application using Visual Studio 2013 and compile/build this application which will create an executable file then run the

".exe" application to execute the code.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RNA\_SeqData\_Analysis

{

class Program

{

static void Main(string[] args)

{

string CHR1Data = string.Empty;

//Read bases from file and make it as string from array which will be used for masking non-exon region.

string[] lines = File.ReadAllLines(@"E:\Bio Informatics\Assignment 8\chr1.fa");

List<string> list = new List<string>(lines);

list.RemoveAt(0);

CHR1Data = string.Join("", list.ToArray());

**////Program - 1\\\\**

//Code below will collapse the exon regions

List<string> FinalData = new List<string>();

string Data = string.Empty;

int Start = 0;

int End = 0;

//Loop thru exon annotation file.

foreach(string ExonData in File.ReadAllLines(@"E:\Bio Informatics\Assignment 8\hg19-refseq-exon-annot-chr1\_sorted"))

{

//Split each line to get details of each exon in exon annotation.

string[] ExonArray = ExonData.Split('\t');

//Check if current start value is same as previous start value then check current end is greater than

//previous end, if so then add entry into list with new end value.

if (ExonArray[1] == Start.ToString())

{

if (Convert.ToInt64(ExonArray[2]) >= End)

{

End = Convert.ToInt32(ExonArray[2]);

}

FinalData.RemoveAt(FinalData.Count - 1);

Data = "chr1\t" + Start.ToString() + "\t" + End.ToString() + "\t" + "X\t" + "0\t+";

FinalData.Add(Data);

}

else if (Convert.ToInt64(ExonArray[1]) <= End)

{

FinalData.RemoveAt(FinalData.Count - 1);

Data = "chr1\t" + Start.ToString() + "\t" + ExonArray[2] + "\t" + "X\t" + "0\t+";//End.ToString()

FinalData.Add(Data);

End = Convert.ToInt32(ExonArray[2]);

}

else

{

if (Convert.ToInt64(ExonArray[1]) >= Start && Convert.ToInt64(ExonArray[1]) <= End)

{

FinalData.RemoveAt(FinalData.Count - 1);

Data = "chr1\t" + Start.ToString() + "\t" + ExonArray[2] + "\t" + "X\t" + "0\t+";//End.ToString()

}

else

{

Data = ExonArray[0] + "\t" + ExonArray[1] + "\t" + ExonArray[2] + "\t" + "X" + "\t" + ExonArray[4] + "\t" + "+";

}

//Store start and end values for further use.

Start = Convert.ToInt32(ExonArray[1]);

End = Convert.ToInt32(ExonArray[2]);

FinalData.Add(Data);

}

}

//Convert list into string and place it into a file.

string CollapsedData = String.Join("\n", FinalData.ToList().ToArray());

File.WriteAllText(@"E:\Bio Informatics\Assignment 8\ExonCollapsedFile.txt", CollapsedData);

**////Program - 2\\\\**

//File will be created or used for writing new chr1 which contains masked non-exon region.

TextWriter tsw = new StreamWriter(@"E:\Bio Informatics\Assignment 8\Masked\_chr1.fa");

tsw.Write(">chr1");

int indexValue = 0;

//Loop through the masked exon region data.

for (int j = 0; j < FinalData.Count; j++)

{

string data = string.Empty;

string[] ChangeData = FinalData[j].Split('\t');

int startIndex = Convert.ToInt32(ChangeData[1]);

//If index value less than start of exon then get substring from already saved chr1 and

//complement with N's. otherwise normal exon data will be picked and placed into a file.

if (indexValue < startIndex)

{

int length1 = startIndex - indexValue;

tsw.Write(ComplementBasesH(CHR1Data.Substring(indexValue, length1)));

}

//Write exon region data.

int endIndex = Convert.ToInt32(ChangeData[2]);

int length2 = endIndex - startIndex;

tsw.Write(CHR1Data.Substring(startIndex, length2));

//Populate end value in index value.

indexValue = endIndex;

}

//Close the text writer.

tsw.Close();

**////Program - 3\\\\**

//Initialize dictionary to store each Exon and its corresponding read count.

Dictionary<string, int> ExonReadData = new Dictionary<string, int>();

//Loop thru originial exon refseq file to find read count for each exon.

foreach(string ExonRefData in File.ReadAllLines(@"E:\Bio Informatics\Assignment 8\hg19-refseq-exon-annot-chr1\_sorted"))

{

string[] ExonRefData\_Array = ExonRefData.Split('\t');

string ExonName = ExonRefData\_Array[3];

//Loop through the bowtie BED file fetched by mapping "ERR030893-1.fq" reads file onto Masked CHR1.

foreach (string BToutData in File.ReadAllLines(@"E:\Bio Informatics\Assignment 8\bowtie-0.12.7\BTout-BED-75\_Modified\_Sorted"))

{

string[] BToutData\_Array = BToutData.Split('\t');

//Check if there exists Exon Name in dictionary and add 1 to the value.

if (ExonReadData.ContainsKey(ExonName))

{

//Add 1 to existing value for particular ExonID. If Map start and end are in range of Exon ID.

if ((Convert.ToInt32(BToutData\_Array[1]) >= Convert.ToInt32(ExonRefData\_Array[1])) &&

(Convert.ToInt32(BToutData\_Array[2]) < Convert.ToInt32(ExonRefData\_Array[2])))

{

ExonReadData[ExonName] = ExonReadData[ExonName] + 1;

}

}

else

{

//Add Exon ID and 1 as initial value. If Map start and end are in range of Exon ID.

if ((Convert.ToInt32(BToutData\_Array[1]) >= Convert.ToInt32(ExonRefData\_Array[1])) &&

(Convert.ToInt32(BToutData\_Array[2]) < Convert.ToInt32(ExonRefData\_Array[2])))

{

ExonReadData.Add(ExonName, 1);

}

}

}

}

//Initialize dictionary for gene level expression by having Gene ID and reads count.

Dictionary<string, int> GeneRead = new Dictionary<string, int>();

//Loop thru list of Exon data.

foreach(KeyValuePair<string,int> ExonRefValue in ExonReadData)

{

string GeneName = ExonRefValue.Key.Substring(0,ExonRefValue.Key.IndexOf("\_e"));

//Check if there exists Gene Name in dictionary and add new value to the existing value.

if(GeneRead.ContainsKey(GeneName))

{

GeneRead[GeneName] = GeneRead[GeneName] + ExonRefValue.Value;

}

else

{

GeneRead.Add(GeneName, ExonRefValue.Value);

}

}

//Convert list into a string and write into a file.

String GeneD = String.Join("\n", GeneRead.ToArray());

GeneD = GeneD.Replace("[", "").Replace("]", "").Replace(',', '\t');

File.WriteAllText(@"E:\Bio Informatics\Assignment 8\GeneID\_read.txt", GeneD);

}

/// <summary>

/// This method is used to complement non-masked region with 'N's.

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

public static string ComplementBasesH(string s)

{

char[] arr = s.ToCharArray();

for (int i = 0; i < arr.Length; i++)

{

arr[i] = 'N';

}

return new string(arr);

}

}

}