An Algorithm for Genomic Classification for Evaluation of Abnormalities

Arun Kumar I A[1*], K.B.Ramesh[1], Vidya Niranjan[2]

[1]Dept. of Electronics and Instrumentation Engineering, RV College of Engineering®, Bengaluru
[2]Dept. of BioTechnology, RV College of Engineering®, Bengaluru

Abstract

SNP (Single Nucleotide Polymorphism) is used to detect complex diseases like pancreatic cancer, age related macular degeneration, prostate cancer and non-Hodgkin lymphoma by extracting the kernel of tag SNP and informative SNP by hierarchical clustering. The detection is also made using random forests for SNP-SNP interaction. Graphic Processing Unit is used for complex data. Energy distribution difference method is used for multiple SNP-SNP interaction and for dealing with different factors of the weighted biological network structure is implemented in turn finding the genome and thereby capturing the complex gene related diseases. The methodology implemented in this research is the hierarchical clustering with the Internet of Things using socket programming for the raw data where the true SNPs are found from the alleles of thymine and cytosine. This is implemented using different forms of raw data analysis where the Protein pattern gets effected by the change in DNA (Deoxyribose Nucleic Acid) with translation and it is used for further analysis of different diseases, specifically rheumatoid arthritis. IoT concept is used to communicate between the physicians, once the result is obtained. The data is stored in the form of cloud and retrieved without any limitations on the data length. Through the internet, an encoding and decoding facility is provided where the data is kept confidential and is open only to the physicians.

Keywords: *Genomic classifications, SNP, Abnormalities, IoT*

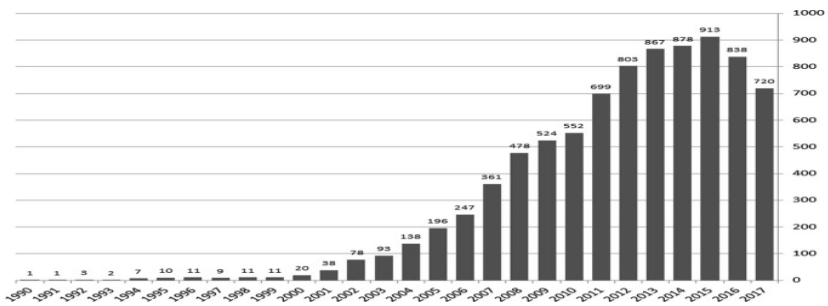
1.0 Introduction

SNP stands for Single Nucleotide Polymorphism. It is found in the body for every thousand nucleotides. There are four to five million nucleotides in the body. The nitrogenous bases attached to the sugar forms nucleoside. Each nucleoside attached to the phosphate group forms nucleotide. These nucleotides are attached with the phosphodiester bond. Technically SNP is also called point mutation [1]. There will be cytosine replaced by the thymine for every thousand nucleotides where these changes will be used for the detection of complex diseases like heart disease, diabetics, cancer which includes pancreatic cancer which includes pancreatic cancer and the inherited diseases. There will be more than 1% of SNP found. According to Bostein and Risch there are altogether one million nucleotides in the human genome [2].

[*]Mail address: Arun Kumar, Student, Department of Electronics and Instrumentation Engineering, RV College of Engineering®, Bengaluru– 59, e-mail: arunkumaria.lbi18@rvce.edu.in

SNPs are located on various parts of the genome with potentially in differing functional implications like within the coding sequences of the genes, non-coding regions of the genes and in an intergenic regional parts. These SNPs do not alter the gene functionalities or protein functions. Consequently, association between the SNP and the developed risk of the tumor or the presence between an SNP and increase in toxicity of an anticancer drug could be due to the dis-equilibrium with another polymorphisms located in the same chromosomal region [3-4]. The existence of an SNP will be indicated only in a proportional of the carriers. In the Oncology, SNP could influence on the susceptibility to specific organ cancer, the clinical result of patients, modify patient's response to the chemotherapy and influence an incidence and the closeness of the side effects produced by the treatment [5]. The existence of the SNPs can be examined from the constitutional DNA which is then drawn from the peripheral blood cells [6]. The distribution of SNP is as shown in the Fig. 1.

Graphical User Interface as shown in Fig. 2 are the data used to get the patterns and are saved in a database. The changes in the DNA by SNP is reflected in the Protein Pattern by translation [7].



X-axis=years, Y-axis=SNP number

Fig. 1. Distribution of SNP disease year-wise

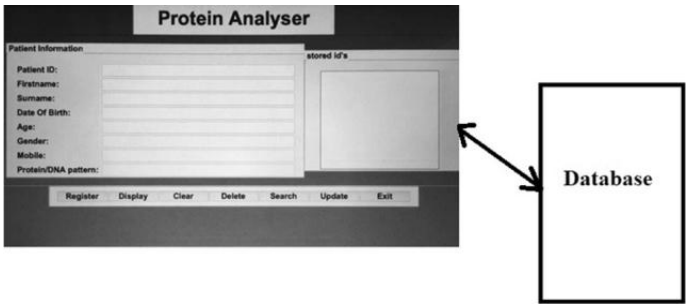


Fig. 2. GUI of Protein analyser

2.0 Design and Implementation

A code is implemented as per the designed flowchart using different modules and in-built functions required for calculating the DNA bases, gathering the standard and obtained result and displaying them in the form of graph.

```
import pandas as _pd
import numpy as _np
import matplotlib.pyplot as _plt
import time

data =
pd.read_csv("http://genome.crg.es/datasets/genomics96/seqs/DNASequences.fa
sta")
data.to_csv("/Users/arunkumaria/Desktop/filename", sep=' ')
print(data)

data1=pd.read_csv("http://genome.crg.es/datasets/sgp2002/testsets/scimit.fa")
data1.to_csv("/Users/arunkumaria/Desktop/filename1", sep=' ')
print(data1)

data2=pd.read_csv("http://genome.crg.es/datasets/gpeval2000/data/SGS/embl50
.h178.masked.fa")
data2.to_csv("/Users/arunkumaria/Desktop/filename2", sep=' ')
print(data2)

print("######Welcome to the World of Bio-Informatics######")

print("\n")
f_obj=open("/Users/arunkumaria/Desktop/filename",'r')
f_obj1=open("/Users/arunkumaria/Desktop/filename1",'r')
f_obj2=open("/Users/arunkumaria/Desktop/filename2",'r')

data=f_obj.read()
str=data.upper()
slen=len(str)

data1=f_obj1.read()
str1=data1.upper()
slen1=len(str1)

data2=f_obj2.read()
str2=data2.upper()
slen2=len(str2)

print("fetching the data containing the DNA....")
time.sleep(3)
print("\n")
print("displaying the data....")
time.sleep(1)
print("\n")
```

```
print(str)#Data Displaying
ac=0
tc=0
gc=0
cc=0
n_count=0
a_count=0

ac1=0
tc1=0
gc1=0
cc1=0

ac2=0
tc2=0
gc2=0
cc2=0
for i in str:
 if(i == 'A'):
  ac=ac+1
 if(i == 'T'):
  tc=tc+1
 if(i == 'G'):
  gc=gc+1
 if(i == 'C'):
  cc=cc+1

for i in str1:
 if(i == 'A'):
  ac1=ac1+1
 if(i == 'T'):
  tc1=tc1+1
 if(i == 'G'):
  gc1=gc1+1
 if(i == 'C'):
  cc1=cc1+1

for i in str2:
 if(i == 'A'):
  ac2=ac2+1
 if(i == 'T'):
  tc2=tc2+1
 if(i == 'G'):
  gc2=gc2+1
 if(i == 'C'):
```

```
 cc2=cc2+1

from pandas import DataFrame

Data = {'x': [ac,tc,gc,cc],
       'y': [ac2,tc2,gc2,cc2]
      }
df = DataFrame(Data,columns=['x','y'])
print (df)

from sklearn.cluster import KMeans

_kmeans = KMeans(n_clusters=2).fit(df)
centroids =_ kmeans.cluster_centers_
print(centroids)

_plt.scatter(df['x'], df['y'], c= _kmeans.labels_.astype(float), s=50, alpha=0.5)
_plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)

char n[10];
structlogin_d
{
        char uname[10];
        char upass[7];
} s;
Int signin(int rd);
Int signup(int rd);
Int signin(int rd)
{
        Charname[10],pass[7];
        Lseek(rd,0,SEEK_SET);
        Printf("\n \t username");
        Fgets(name,10,stdin);
        Printf("\n \t password");
        Fgets(pass,7,stdin);
        Printf("\n \t password");
        Print("\n\n");
        Lseek(rd,0,SEEKSET);
        While(read(rd,&s,sizeof(s)))
        {
                If(strcmp(s.uname.name)==0)
                {

                        If(strcmp(s.upass.pass)==0)
                        {
```

```
                                        Strcpy(n,name);
                                        Return 1;
                                }
                        }
                }
        }
        Return 0;
}

Int signup(int rd)
{
        Char name[10],Pass[7],name[20];
        Printf("\nenter required username");
        //fpurge(stdin);
        Fgets(name,10,stdin);
        Printf("enter the password for your profile");
        //fpurge(stdin);
        Fgets(pass,7,stdin);
        Strcpy(s.uname,name);
        Strcpy(s.upass,pass);
        Write(rd,&s,sizeof(s));
        Name[0]='\0';
        Strcpy(nam,name);
        Strcat(nam,"serverfile");
        Open(nam,O_CREAT|O_RDWR|O_APPEND,0777);
Nam[0]='\0';
        }
}
##server##
int main()
{
        char ch2;
        int sd=open("server_data",O_RDWR |O_APPEND,0666);
        int sid=socket(PF_INET,SOCK_STREAM,0);

        int rd = open("security",O_WRONLY | O_CREAT, 0666);
        printf("\n\n\n==========================WELCOME TO THE
DATA SERVER==================================\n\n\n");
        printf("\n==>Assign an account for a CLIENT to transfer a FILE to the
SERVER.....\n");
        signup(rd);
        if(sid == -1)
        {
                perror("Socket");
                exit(0);
        }
```

```
	else
	{
		struct sockaddr_in ser_sock;
		ser_sock.sin_family = AF_INET;
		ser_sock.sin_port = htons(5051);
		ser_sock.sin_addr.s_addr = inet_addr("127.0.0.1");
int b = bind(sid,(struct sockaddr *)(&ser_sock),sizeof(ser_sock));
		if(b == -1)
		{
			perror("Bind");
			exit(0);
		}
		else
		{
			while(1)
			{
				int l = listen(sid,2);
	if(l==-1)
				{
					perror("Listen");
					exit(0);
				}
				else
				{
					struct
sockaddr_in cl_sock;
int size = sizeof(cl_sock);
					int cid = accept(sid,(struct sockaddr
*)(&cl_sock),&size);				if(cid  == -1)
					{
				perror("accept");
				exit(0);
					}
					else
					{
printf("Acknowledgement Successful...!!!\n");
printf(">>>>>>>Accepting the FILE from the CLIENT.....\n\n\n");
sleep(10);
	while(read(cid,&ch2,1))

					{
						write(sd,&ch2,1);
					}
						printf("*****FILE received
Successfully....!!!\n");
```

```
printf("==>Assign an account for a CLIENT to transfer another FILE to the
SERVER.....\n");
                                                signup(rd);
                                        }

                                }
                        }
                }
        }
}
##client##
int main(int argc,char *argv[])
{
        int _eq;
        char _ch[60];
        //int nd=open("client_file1",O_RDWR);
        int nd=open("client_data1",O_RDONLY|O_CREAT,0666);
        char ch1;
        int rd = open("security",O_RDONLY);
        printf("\n==>Verify your account to transfer a FILE to the
SERVER....\n");
        eq = signin(rd);
        if(eq == 1)
        {
                int cid = socket(PF_INET,SOCK_STREAM,0);
                if(cid == -1)
                {
                        perror("Socket");
                        exit(0);
                }
                else
                {
                        struct sockaddr_in ser_sock;
                        ser_sock.sin_family = AF_INET;
                        ser_sock.sin_port = htons(5051);
                        ser_sock.sin_addr.s_addr = inet_addr("127.0.0.1");
int c = connect(cid,(struct sockaddr *)(&ser_sock),sizeof(ser_sock));
                        if(c==-1)
                        {
                                perror("connect");
                                exit(0);
                        }
                        else
                        {
                                while(read(nd,&ch1,1))
```

```
                                        {
                                            write(cid,&ch1,1);
                                        }
                                    }
                                }
                        }
                        else
                        {
                                printf("\n you are not the user \n");
                        }
}
/*end of the server client coding lines*/
```

3.0 Methodology

The program code works like a database. Once the data is extracted it is categorized and a graph is plot with respect to the present reference data. From the obtained graph one can identify the cytosine variations [8] and use it to debug the disease in the body. The obtained graph uses the K-means clustering. The libraries like pandas, numpy, matplotlib are used for the data extraction, representing the array and to get the result in the form of graphs. Data set of readily available and to be extracted into the csv format works successfully. Socket Programming [9] is used to send the live results to the remote physicians for analyzing the data using encryption and decryption methodologies.

4.0 Functional Block diagram and Flowchart

Fig. 3 consists of the raw form of data which is stored in a temporary file and then retrieved. It is then gathered as adenine (A), guanine(G), thymine(T), cytosine(C). The graph is plotted by clustering method [10] by taking the standardized AGTC count and the obtained AGTC count.

In Fig. 4, the obtained and analysed result is sent through socket programming to the remote physician by encoding and decoding through internet where IoT concept [11] is being involved to store the data and retrieve it efficiently. The data hence sent is preserved and retrieved without any limits [12].

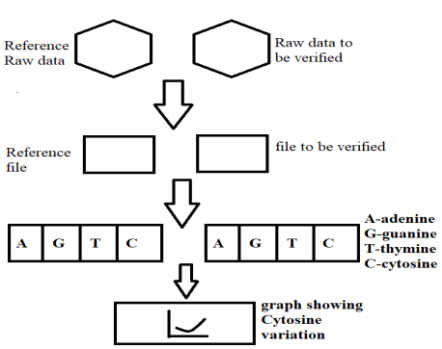


Fig. 3. Flowchart of a functional block

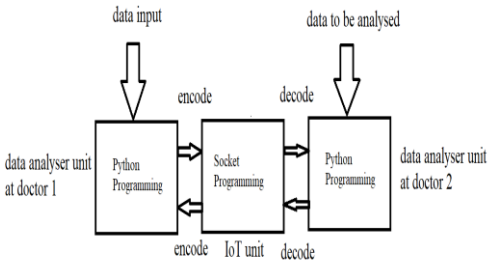


Fig. 4. Flowchart of an IoT based connection

5.0 Results

The standard data is compared with the obtained calculated data to get the graph of following pattern. The importance of SNP in the disease identification is mainly highlighted. For identifying the disease here from coding using Python we have the different sorts of data where the presence of cytosine is seen in which we are able to get the result of cytosine replaced by thymine instantly represented by the diagram using the K-means clustering where in case the missed alleles are found. Plenty of diseases like cancer, sickle-cell anaemia, beta-thalassaemia and cystic fibrosis are identified.

	X(standard in counts)	Y(calculated in counts)
adnine	736902	245056
guanine	763077	252562
thymine	702601	253079
cytosine	689613	248551

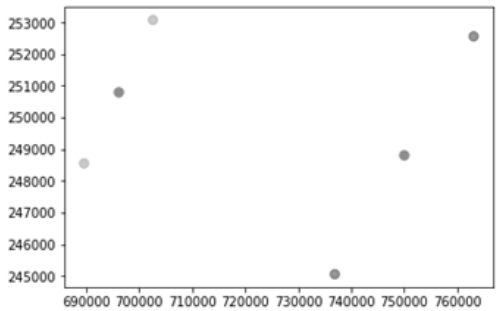


Fig. 5. Distribution of adenine, guanine, thymine and cytosine

Fig. 5 shows the distribution of adenine, guanine, thymine, cytosine for a given set of data. Yellow colored data shows the replacement of the alleles, i.e.., thymine and cytosine. Red colored data is the centroid point. Blue colored data is the distribution of adenine & guanine.

Conclusions

Importance of SNP in the disease identification is mainly highlighted. For identifying the disease based Python coding, different sorts of data are obtained

in which the presence of cytosine is observed. Cytosine replaced with thymine instantly as represented in the diagram using the K-means clustering in which missed alleles are found. Diseases like cancer, sickle-cell anemia, beta-thalassaemia and cystic fibrosis can be identified using the algorithm developed. Future enhancements can be made to directly analyze the graph with different diseases by assigning the threshold which help to segregate the diseases more clearly.

References

1. X Li, B Liao, L Cai, Z Cao, W Zhu, Informative SNPs selection based on two-locus and multilocus linkage disequilibrium: Criteria of max-correlation and min-redundancy, *IEEE/ ACM Trans. Comput. Biol. Bioinformat*, 10 (3), 688–695, 2013
2. P I De Bakker, R R Graham, D Altshuler, B E Henderson, C A Haiman, Transferability of tagSNP stoc apture common genetic variation in DNA repair genes across multiple populations, *in Proc.Pac.Symp.Biocomput*, 11, 478–486, 2006
3. C S Carlson, M A Eberle, M J Rieder, Q Yi, L Kruglyak, D A Nickerson, Selecting a maximally informative set of single nucleotide polymorphisms for association analyses using linkage disequilibrium, *Amer. J. Human Genetics*, 74 (1), 106– 120, 2004
4. R C Hardison, G A Blobel, GWAS to therapy by genome edits?, *Science*, 342 (6155), 206–207, 2013
5. W Yang, C C Gu, A whole-genome simulator capable of modeling high-order epistasis for complex disease, *Genetic Epidemiol*, 37 (7), 686–694, 2013
6. A Gyenesei, J Moody, C A Semple, C S Haley, W H Wei, High-throughput analysis of epistasis in genome-wide association studies with BiForce, *Bioinformatics*, 28 (15), 1957–1964, 2012
7. B Liao, X Li, W Zhu, R Li, S Wang, Multiple ant colony algorithm method for selecting tag SNPs, *J. Biomed. Informat*, 45 (5), 931–937, 2012
8. C K Ting, W T Lin, Y T Huang, Multi-objective tag SNPs selection using evolutionary algorithms, *Bioinformatics*, 26 (11), 1446–1452, 2010
9. J He, A Zelikovsky, Informative SNP selection methods based on SNP prediction, *IEEE Trans. Nanobiosci*, 6 (1), 60–67, 2007
10. Z Q Liu, S L Lin, Multilocus LD measure and tagging SNP selection with generalized mutual information, *Genetic Epidemiol*, 29 (4), 353–364, 2005
11. Alasdair Gilchrist, Industry 4.0, *IoT*, 29 (4), 353–364, 2019
12. Alp Ustundug, Emre Cevikcan, Industry 4.0 managing the digital transformation, *Cloud computing*, 44 (4), 353–364, 2019