19AIE103_Biology

PROJECT TOPIC: GENE PREDICITON AND EDITING

B. Tech (CSE (AI)) (2020 Batch)

Team number: 7

AIEA.12026	[Dinesh Kumar M R]
AIEB.29621	[Divi Eswar Chowdary]
AIEB.35658	[Dabbara Harsha]
AIEB.23682	[B.E. Pranav Kumar]

Acknowledgement:

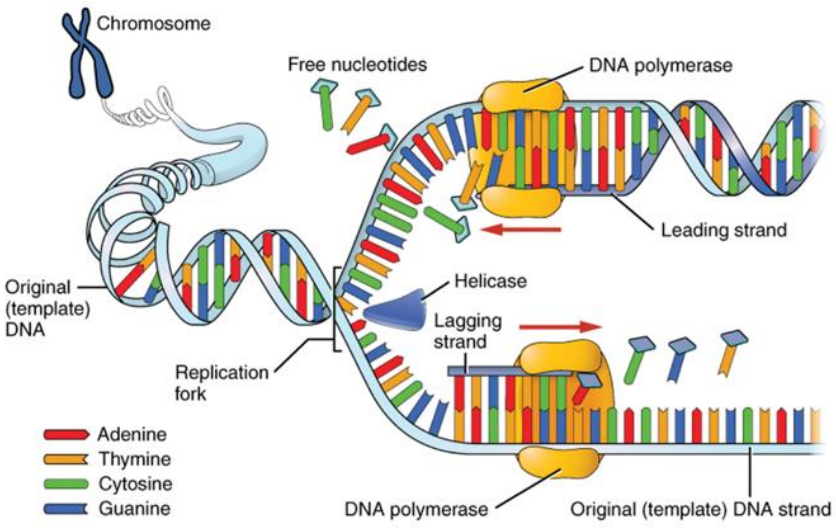
We would like to express our special thanks of gratitude to our teacher Dr. Geetha who gave us the golden opportunity to do this wonderful project on the topic (GENE PREDICITON AND EDITING), which also helped us in doing a lot of Research and we came to know about so many new things. We are thankful for the opportunity given.

Contents

Introduction:

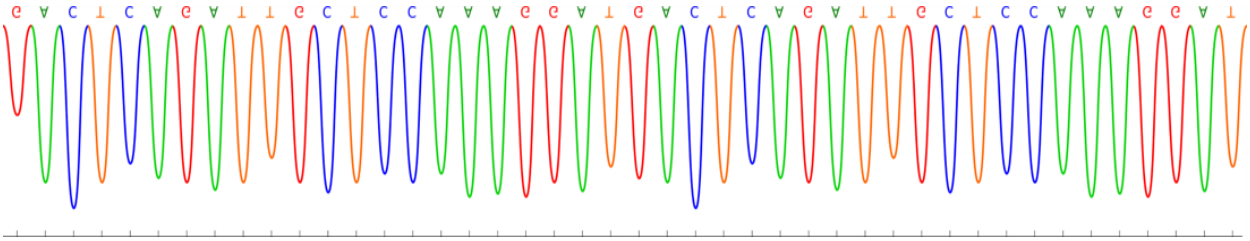
To understand genes, we need to understand some related concepts such as DNA and DNA sequencing. Deoxyribonucleic acid (DNA) is a molecule composed of two polynucleotide chains that coil around each other to form a double helix carrying genetic instructions for the development, functioning, growth and reproduction of all known organisms and many viruses. The two DNA strands are known as polynucleotides. Each nucleotide is composed of one of four nitrogen-containing nucleobases (cytosine [C], guanine [G], adenine [A] or thymine [T]), a sugar called deoxyribose, and a phosphate group.

These chains contain vital information in their arrangement which allows cells to use them like a manual. The two strands of DNA run in opposite directions to each other and are thus antiparallel. This information is replicated as and when the two strands separate. This DNA is replicated during reproduction of the cell and an equal amount of which is specific to each organism is always maintained. The following image will give us a rough idea,

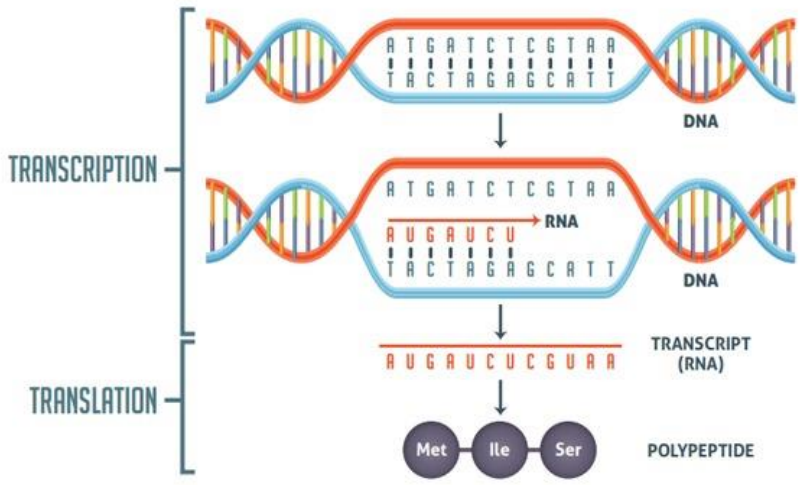


DNA sequencing is the process of determining the nucleic acid sequence – the order of nucleotides in DNA. It includes any method or technology that is used to determine the order of the four bases: adenine, guanine, cytosine, and thymine. The advent of rapid DNA sequencing methods has greatly accelerated biological and medical research and discovery.

The process is automated where different fluorescently labelled nucleotides are used to bind to the nucleotides of the DNA fragments and the fluorescent signal emitted from the machine are recorded by the machine. The output signal of each fluorescent signal is represented in terms of a peek. Each "peek" represents each fluorescently labelled nucleotide binds with the DNA sequence. After completion of the process, the data of genome sequencing are sent for the bioinformatics analysis. The processed result looks like this,



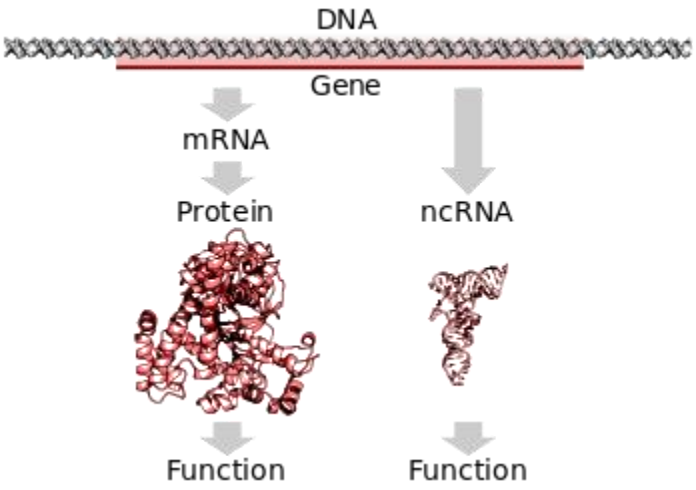
How this is relevant?



RNA strands are created using DNA strands as a template in a process called transcription, where DNA bases are exchanged for their corresponding bases except in the case of thymine (T), for which RNA substitutes uracil (U). Now we have obtained a corresponding RNA strand/s. Under the genetic code, these RNA strands specify the sequence of amino acids within proteins in a process called translation. With this we have obtained the proteins that build the body of the organism. These proteins are directly responsible for the traits that the organism possesses. If we look at the order in reverse, we see that DNA is fundamentally the reason for traits.

What is a gene?

The strands of DNA are effectively noted to be divided into segment/s or stretch/s which encodes for different traits. These are just unique selections of the DNA strands and hence they also follow the same two processes (transcription and translation) to make the proteins. Their effects reflect on the phenotype.



Consequently, all traits are accounted for by the organism's genes. These are different for each organism which end up giving the organism its unique traits.

Over years the genes are combined in multiple ways due to reproduction and some diversity is formed in the gene pools of the organism. Gene diversity is a desirable factor as it accounts for the survival of the species. Sometimes during reproduction these genes are wrongly recombined or the organism is exposed to harmful DNA damaging events like radiation, chemical etc. If the organism does not die due to these changes (if the gene was repaired but not correctly) and these genes are passed down by heredity. These in turn most probably negatively affect the organism.

We will look at how we choose to get rid to these errors,

Gene identification:

Why is there a need to predict genes?

Naturally if we can identify a gene that is responsible for a phenotype then we can use that information to edit the gene to better suit us. (this tells us where to look for and what to change to get desired results)

How can this be done?

- First, we will have to go through the tedious task of collecting and organizing the data. (forming a data base)
- In this case the data is relationships between the gene responsible for the phenotype of interest.
- Then we must compare and evaluate the data in order to pin-point the gene correctly (why is this important? Ans: if wrongly done it may cause more harm than good to the subject)
- This part of the process is time consuming and if the data set is complex or large, this might not be humanly possible within a lifetime.

How can the data base be created?

- This slide describes steps involved in the process of DNA sequencing in brief:

1. Sample collection: any DNA sample is collected to be processed on.
2. DNA extraction: Isolating DNA by disrupting cell wall/cell membrane and a nuclear membrane is called a DNA extraction.
3. Quantification and purification of DNA: the inhibitors present in the sample may hurdle in sequencing.
4. DNA fragmentation and library preparation: Digest the DNA sample using various endonucleases for creating a library or cloning the DNA chunk. (this process is unique to the method used in terms of endonucleases)

What other methods are there to reliably compare the data?

- We can use computers to aid us in this process.

In our project we have coded an algorithm in python using excel files as a database which allows us to correctly predict the gene in the given dataset that is responsible for the phenotype

Code logic:

- It selects a true phenotypes sequence and uses it as a basis for comparison. (what does this do? It ensures that if the phenotype is truly genetic then a completely true value accommodates for it)

- What if it is not true? in that case a user must make a small alteration in the data base to make this code relevant (swap a false value for a true one as the first sequence)
- The length of the gene used in comparison is custom (user set) (what does this do? Gives the user options)

Comparison logic

- We iterate and select a specified sized comparison gene sequence
- We vertically traverse through the sequences (compares with all sequences)
- During each sequence which we horizontally compare the sequences with the selected gene segment.
- With the comparison data we then evaluate it

Evaluation logic

- This is based on how a human would procedurally conduct the process.
- If the phenotype and the gene sequence match, then we keep track of this by a variable in the code named "count true" (what does this represent? The gene is responsible for the observed phenotype: favorable outcome)
- If the phenotype does not match but the gene sequence matches, then we keep track of this by a variable in the code named "count false" (what does this represent? The gene is not responsible for the observed phenotype, a shared gene (accounts for randomness): unfavorable outcome)
- All other outcomes are also accounted for (why? Because we are looking at probability, helps rank a gene if data set is small)
- We remove the unfavorable form the favorable and give a probability score for each compared segment

- After we get the probabilities of each code, we then look for the most probable gene

Python code showing application of logics

For comparison logic:

```
for n in range(1,rowmax-1):
    selements = range((sdnawidth*n),sdnawidth*(n+1),1)
    #print(selements)
    for indx in selements:
        temp.append(geneinfo[indx])
    tempst = ''.join(temp)
    tempstf = tempst.upper()
```

For evaluation logic:

```
counttrue = 0
countfalse = 0
eval = tempstf.find(compstf)
    if phenotype[n] == 't':
        if eval>=0 and eval<sdnawidth:
            counttrue += 1
    else:
        if eval >= 0 and eval < sdnawidth:
            countfalse +=
```

Integrating these logics to get one single functional code.

Entire code file is in the respective folder.

Features of this code:

- It solves the basic purpose that is to order the trail set and is faster than conventional methods.
- It is simple and clean which makes it extremely customizable (new features can be added without risking the integrity of the preexisting code) [with respect to the need we can even modify the purpose of this code]
- User can set the length of the comparison gene.
- User can select the excel to work with.
- Error handling: the following errors have been accommodated for

I. Size of the matrix

II. Text fonts and case of data in database

III. No manipulation during runtime

Limitations of this code:

- This code cannot show the evidence of comparison in visual form to viewer (why is visual evidence important? It allows humans to be certain that it has followed the correct process and when the pattern is pointed out we can easily verify the effectiveness of the program, makes the code more "reassuring to the user")
- This code is not associated with a GUI all programs without a complimentary GUI don’t make it far in the market.
- the database but cannot function with data type or data errors. If the database contains any false or missing data, it cannot adapt accordingly.

- As pointed out earlier, the user must enter the all data with at most precision. (but that should be the feature of the database filling program.)
- The first data input must be that of a true phenotype.
- In order to produce a more flexible code evaluation formula we must have some biological relevance.
- What can be done? We look to overcome these limitations or improve this code soon.

Verification of our code:

1.We ran the code using some preset data with selected patterns to check for the right functionality.

Demo.xlsx - contains predefined data with intentionally created

	A	B	C	D	E	F	G
1	subject no	phenotype	p1	p2	p3	p4	p5
2	1	t	A	T	G	C	C
3	2	t	C	C	G	T	A
4	3	f	G	A	C	A	T
5	4	t	A	C	C	T	G

Corresponding experiment:

Input parameters:

```
enter the file name(dataset name) :demo.xlsx
enter the number of individual segments we have to compare :2
```

Result:

```
[-8.33, 8.33, 0.0, 16.67]
16.67
3
['AT', 'TG', 'GC', 'CC']
the gene segment most probable to have caused the phenotype is CC

Process finished with exit code 0
```

2.

Demo1.xlsx - additional data (in terms of no of observations)

The same pattern was maintained for ease of observation.

	A	B	C	D	E	F	G
1	subject no	phenotype	p1	p2	p3	p4	p5
2	1	t	A	T	G	C	C
3	2	t	C	C	G	T	A
4	3	f	G	A	C	A	T
5	4	t	A	C	C	T	G
6	5	t	T	G	C	C	A
7	6	f	C	T	A	G	C

Corresponding experiment:

Input parameters:

```
enter the file name(dataset name) :demo1.xlsx
enter the number of individual segments we have to compare :2
```

Result:

```
[-5.0, 10.0, 0.0, 15.0]
15.0
3
['AT', 'TG', 'GC', 'CC']
the gene segment most probable to have caused the phenotype is CC

Process finished with exit code 0
```

3.

Demo2.xlsx - additional data (in terms of length of DNA)

The same pattern of exp 1 was maintained but length of DNA sequence was maintained.

	A	B	C	D	E	F	G	H	I
1	subject no	phenotype	p1	p2	p3	p4	p5	p6	p7
2	1	t	A	T	G	C	C	T	A
3	2	t	C	C	G	T	A	G	C
4	3	f	G	A	C	A	T	C	G
5	4	t	A	C	C	T	G	A	t

Corresponding experiment:

Input parameters:

```
enter the file name(dataset name) :demo2.xlsx
enter the number of individual segments we have to compare :2
```

Result:

```
[0.0, 5.56, 5.56, 11.11, 5.56, 5.56]
11.11
3
['AT', 'TG', 'GC', 'CC', 'CT', 'TA']
the gene segment most probable to have caused the phenotype is CC

Process finished with exit code 0
```

4.

Demo3.xlsx - combination of both 1 and 2

	A	B	C	D	E	F	G	H	I
1	subject no	phenotype	p1	p2	p3	p4	p5	p6	p7
2	1	t	A	T	G	C	C	T	A
3	2	t	C	C	G	T	A	G	C
4	3	f	G	A	C	A	T	C	G
5	4	t	A	C	C	T	G	A	t
6	5	t	T	G	C	C	A	c	g
7	6	f	C	T	A	G	C	g	a

Corresponding experiment:

Input parameters:

```
enter the file name(dataset name) :demo2.xlsx
enter the number of individual segments we have to compare :2
```

Result:

```
[0.0, 5.56, 5.56, 11.11, 5.56, 5.56]
11.11
3
['AT', 'TG', 'GC', 'CC', 'CT', 'TA']
the gene segment most probable to have caused the phenotype is CC

Process finished with exit code 0
```

The following experiments verify the integrity of the code under the limitations.

Gene editing

- Without realizing we have been doing this by selective breeding for a long time. Ever since we have managed to understand DNA and its purpose, we have replaced it with more sophisticated methods.
- Now that we have managed to get an optimized gene segment responsible for the observed phenotype and its location, we look for suitable replacement of this gene with one with the following properties:

i. Must be compatible with old DNA

ii. Must give a beneficial trait or remove the disadvantage of the previous one

In some cases, just deleting a gene segment will do to remove the drawbacks that come with it.

This can be done by experimenting in reference to data gained form previous experiments by performing the following process:

How can gene editing be done?

1.Double strand break repair:

A common form of Genome editing relies on the concept of DNA double stranded break (DSB) repair mechanics. There are two major pathways that repair DSB; non-homologous end joining (NHEJ) and homology directed repair (HDR). NHEJ uses a variety of enzymes to directly join the DNA ends while the more accurate HDR uses a homologous sequence as a template for regeneration of missing DNA sequences at the break point. This can be exploited by creating a vector with the desired genetic elements within a sequence that is homologous to the flanking sequences of a DSB. This will result in the desired change being inserted at the site of the DSB. While HDR based gene, editing is like the homologous recombination-based gene targeting, the rate of recombination is increased by at least three orders of magnitude.

2.Engineered nucleases:

The key to genome editing is creating a DSB at a specific point within the genome. Commonly used restriction enzymes are effective at cutting DNA, but generally recognize and cut at multiple sites. To overcome this challenge and create site-specific DSB, three distinct classes of nucleases have been discovered and bioengineered to date. These are the Zinc finger nucleases (ZFNs), transcription-activator like effector nucleases (TALEN), mega nucleases and the clustered regularly interspaced short palindromic repeats (CRISPR/Cas9) system.

The most promising among these gene editing methods is CRISPR because of it is cheap (99%), fast (weeks opposed to years) and precise

CRISPR

- By observing the way bacteria remember a part of the bacteriophage and later use the same gene given to an enzyme cas9 to cut the virus rendering it powerless and realizing that the CRISPR is programable we can pretty much outsource editing to a naturally existing system.
- We can just give CRISPR a segment of DNA that is to be modified and put the system into a living cell.
- Another advantage is that it is extremely precise as it must find a 100% match in the sequence to edit it.
- Newer methods are in development at present.

Applications

- Can help eradicate genetic diseases
- Can genetically engineer embryos

- Can modify immune cells to become better cancer hunters
- Can make plants more pest resistant, better yielding and more robust in extreme conditions
- Can engineer animals to better adapt to human needs

Bibliography:

Definitions:

https://en.wikipedia.org/wiki/

https://www.google.com/

Images:

https://www.google.com/

Referred articles:

Genetic Engineering Will Change Everything Forever – CRISPR

https://en.wikipedia.org/wiki/DNA

https://en.wikipedia.org/wiki/DNA_sequencing

Read And Write Excel Files In Python Using Openpyxl In PyCharm-Excel Styling and formatting Python

Thank you