Herramienta para el análisis de las proteínas que interactúan con LASP-1

Repositorio: https://github.com/Fabio-Coronado/Proyecto_BC

Fabio Coronado, Rodrigo Paz

October 26, 2019

Abstract

The proposal for this project is to carry out a graphic interface that facilitates the analysis of the proteins that interact with LASP-1 and its association with different types of carcinomas, especially hepatocellular carcinoma related to HBV, in addition to carrying out a phylogenetic analysis of these proteins that It will allow us to locate its evolutionary origin.

For phylogenetic analysis we will look for their respective orthologs that will help us find the origin of these proteins.

1 Resumen Ejecutivo

La propuesta para este proyecto es realizar una interfaz gráfica que facilite el análisis de las proteínas que interactúan con LASP-1 y su asociación con distintos tipos cárcinomas en especial el carcinoma hepatocelular relacionado con el VHB, además de realizar un ánalisis filogenético de estas proteínas que nos permitirá ubicar su origen evolutivo.

Para el análisis filogenético buscaremos sus respectivos ortólogos que nos ayudarán a encontrar el origen de estas proteínas.

2 Descripción del proyecto

Para este proyecto se toma en cuenta un estudio realizado que está mencionado en la bibliografía. El estudio realiza el siguiente análisis:

- Para encontrar las proteínas que interactuan con LASP-1, se recoge información de proteínas humanas en distintas base de datos. Una vez encontradas las proteínas se descartan las proteínas que no son validadas experimentalmente.
- Luego se evalúa la clase de proteína de los interactuadores LASP-1 utilizando el sistema de clasificación PANTHER. Se realiza la anotación de la función génica de los interactuadores LASP-1 con el análisis GO.
- Las rutas en las que los interactores LASP-1 están involucrados son evaluadas por la ruta KEGG y la ruta PANTHER, se construye la red de interacción que contiene nodos correspondientes a LASP-1 con sus proteínas interactuantes y se integra aún más la información de interacción de interactuadores LASP-1 con sus rutas asociadas.
- Como se sabe de la sobreexpresión de LASP-1 en HCC, y se asocia con infección por VHB en pacientes con CHC, se investigó si los interactuadores LASP-1 también están asociados con CHC relacionado con VHB.
- Basado en el análisis de perfiles de expresión génica de GSE14520, se encontró que el cambio en LASP-1 fue mayor de

2.0 veces con un P-valve < 0.05 en tejidos HBV-HCC en comparación con tejidos no HCC como se muestra en la Figura 1.

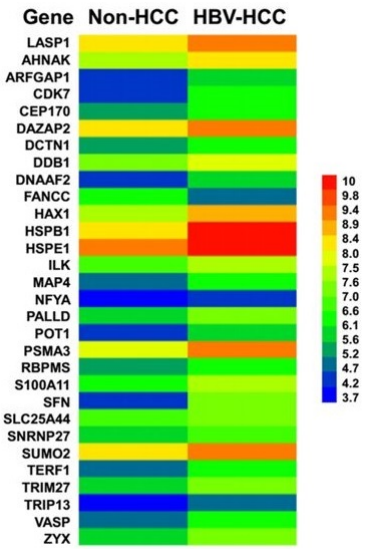

Figure 1: La expresión relativa de los interactuadores LASP-1 con un cambio de pliegue de aproximadamente 2 y $P < 0.05$

- Entre estos genes, solo FANCC esta regulado negativamente en el HCC relacionado con el VHB, y los niveles de expresión de otros 28 genes aumentaron en las células enfermas.

Escogeremos las siguientes proteínas de las 28 que aumentaron su expresión, estas serán:

- LASP-1
- ZYX
- HSPB1
- SFN
- HAX1
- SUMO2
- ILK

Al realizar una investigación en distintas áreas muchas veces el limitante es el financiamiento, para ayudar quitar costos hemos optado en este proyecto por crear un software que utiliza recursos gratuitos como librerías de python para la parte bioinformática en las que se incluirán : biopython, biopandas y para el GUI utilizaremos Tkinter.
Otro factor que puede ser limitante es el hardware disponible, por lo que en lugar de descargar las bases de datos, realizaremos consultas a las base de datos en línea mediante el programa.

Las herramientas que serán implementadas en el software (GUI) serán:

- Alineamiento global de secuencias.
- Alineamiento local de secuencias.
- Alineamiento múltiple de secuencias.
- Algoritmos de búsqueda de secuencias en Base de datos.
- Algoritmo BLASTp.
- Algotirmo para el análisis filogenético.

Se incluirán distintos algoritmos de programación dinamica: como los algoritmos de alineamiento de secuencias usando una matriz BLOSUM62 (puede ser cambiada), Algoritmo Blast para la búsqueda de secuencias similares en una base de datos, Algoritmo CLUSTALW para el alinemaiento múltiple.

Se entregarán los avances de acuerdo al siguiente cronograma:

Entregable 1(Practica Calif N.2.:)

- Propuesta del Proyecto.

Entregable 2(Práctica Calif N.3.):

- Versión final de la propuesta en PDF.
- Proteínas escogidas que interactúan con LASP-1.
- Avance del proyecto al 30%.

Entregable 3(Práctica Calif N.4.):

- Implementación al 70%.
- Cálculo de descriptores de interacción proteína - proteína con el software.

Entregable 4 (Lab. Calif. N.4):

- Implementación al 100%. Incluye: Código fuente, Librerías, datos de prueba, reporte final. (DVD o repositorio público).
- Análisis y Generación de Árboles Filogenéticos.

Aporte personal de cada integrante:

- Fabio Coronado:
  - Análisis filogenético.
  - Algoritmos de búsqueda de secuencias en Base de datos.
  - Alineamiento de secuencias.
- Rodrigo Paz:
  - Implementación del GUI en Tkinter.
  - Algoritmo BLASTp.
  - Alineamiento múltiple de secuencias.

3 Algoritmos e Implementación Computacional

En el proyecto se implementŕan distintos algoritmos que tienen un costo computacional adecuado para el análisis de una gran cantidad de datos.

Algunos algoritmo ya implementados para esta presentación:

Algoritmo de Needleman-Wunsch:

```
def needleman_Wunsch(seq1,seq2,sm,g):
S = [[0]]
T = [[0]]
# initialize gaps row
for j in range(1, len (seq2)+1):
S[0].append(g * j)
T[0].append(3)
# initialize gaps column
for i in range(1, len (seq1)+1):
S.append([g * i])
T.append([2])
# apply the recurrence relation
#to fill the remaining of the
#matrix
for i in range(0, len (seq1)):
for j in range( len (seq2)):

s1=S[i][j]+score_pos(seq1[i],seq2[j],sm,g);
s2 = S[i][j+1] + g
s3 = S[i+1][j] + g
S[i+1].append(max(s1, s2, s3))
T[i+1].append(max3t(s1, s2, s3))
return (S, T)
def max3t (v1, v2, v3):
if v1 > v2:
if v1 > v3: return 1
else : return 3
else :
if v2 > v3: return 2
else : return 3
```

Algoritmo de Smith-Waterman:

```
def smith_Waterman (seq1, seq2, sm, g):
S = [[0]]
T = [[0]]
maxscore = 0
for j in range(1, len (seq2)+1):
  S[0].append(0)
  T[0].append(0)
for i in range(1, len (seq1)+1):
  S.append([0])
  T.append([0])
for i in range(0, len (seq1)):
  for j in range( len (seq2)):
      s1 = S[i][j] +
          score_pos(seq1[i], seq2[j], sm, g)
      s2 = S[i][j+1] + g
      s3 = S[i+1][j] + g
      b = max(s1, s2, s3)
      if b <= 0:
        S[i+1].append(0)
        T[i+1].append(0)
      else:
        S[i+1].append(b)
```

```
        T[i+1].append(max3t(s1, s2, s3))
        if b > maxscore:
            maxscore = b
return (S, T, maxscore)
```

Algoritmo para obtener el alineamiento:

```
def recover_align (T, seq1, seq2):
res = ["", ""]
i = len (seq1)
j = len (seq2)
while i>0 or j>0:
i f T[i][j]==1:
res[0] = seq1[i-1] + res[0]
res[1] = seq2[j-1] + res[1]
i -= 1
j -= 1
e l i f T[i][j] == 3:
res[0] = "-" + res[0]
res[1] = seq2[j-1] + res[1]
j -= 1
e l s e :
res[0] = seq1[i-1] + res[0]
res[1] = "-" + res[1]
i -= 1
return res
```

Alineamiento múltiple de secuencias en Biopython:

```
from Bio import Alphabet
from Bio.SeqRecord import SeqRecord
from Bio.Align import MultipleSeqAlignment
from Bio.Alphabet import IUPAC
from Bio.Seq import Seq
seq1="MHQAIFIYQIGYPLKSGYIQSIRSPEYDNW"
seq2="MH--IFIYQIGYALKSGYIQSIRSPEY-NW"
seq3="MHQAIFI-QIGYALKSGY-QSIRSPEYDNW"

seqr1=SeqRecord(Seq(seq1,
Alphabet.Gapped(IUPAC.protein)),id ="seq1")
seqr2=SeqRecord(Seq(seq2,
Alphabet.Gapped(IUPAC.protein)),id ="seq2")
seqr3=SeqRecord(Seq(seq3,
Alphabet.Gapped(IUPAC.protein)),id ="seq3")

alin=MultipleSeqAlignment([seqr1, seqr2, seqr3])
print(alin)

#diferentes formas de imprimir:
# 2nd sequence
print (alin[1])
# 3rd column
print (alin[:,2])
# 4th to 7th columns (all sequences)
print (alin[:,3:7])
# first 3 columns of seq1
print (alin[0].seq[:3])
# sequences 2 and 3; 4th to 10th column
print (alin[1:3,5:12])
```

Algoritmo para usar Blastp:

```
from Bio.Blast import NCBIWWW
from Bio.Blast import NCBIXML
from Bio import SeqIO
import re
#guarda un archivo xml
def Blast(archivofasta, direccion
                        ,nombrearchivo):

record = SeqIO.read(open(archivofasta),
                            format="fasta")

result_handle = NCBIWWW.qblast("blastp",
                "nr", record.format("fasta"))
save_file = open( direccion+"/"+
                    nombrearchivo+".xml", "w")
save_file.write(result_handle.read())
save_file.close()
result_handle.close()
```

4 Resultados

- Esperamos encontrar el origen evolutivo de las proteinas escogidas.

- Además esperamos demostrar con el árbol filogenético el origen de LASP-1, ya sabemos de antemano que algunos análisis indican que puede tener origen en la evolución de invertebrados a vertebrados.

5 Conclusiones

- En un futuro se podría implementar mas funcionalidades el software como: Modelamiento de proteinas con PyMol, Algoritmos ocultos de Markov, Motif Discovery, etc.

- Conocer el origen evolutivo de estas proteínas nos permitirá estar más cerca de encontrar el origen de distintos tipo de cáncer, esto tal vez ayudará a encontrar tratamientos más efectivos.

- Con el pasar del tiempo se descubren diferentes proteínas que interactúan con carcinomas, encontrar las proteínas correctas mejorará el diagnóstico de los distintos tipos de cáncer.

- Al ser una herramienta que incluye librerías libres permite ahorrar costos en la investigación.

References

[1] Kong Fan-Yun, Zhu Ting, Li Nan, Cai Yun-Fei,Zhou Kai,Wei Xiao, Kou Yan-Bo, You Hong-Juan, Zheng Kui-Yang,Tang Ren-Xian
Analysis of the proteins interacting with LASP-1 and their association with HBV-related hepatocellular carcinoma, 03 March 2017,
https://www.nature.com/articles/srep44017

[2] Alan D. Moore.
Python GUI Programming with Tkinter, May 2018

[3] Miguel Rocha , Pedro G. Ferreira.
Bioinformatics Algorithms, Design and Implementation in Python, 2018

[4] Tiago Antao.
Bioinformatics with Python Cookbook - Second Edition, 2018