RESEARCH

# A sample article title

Jane E. Doe* and John R.S. Smith

*Correspondence:
jane.e.doe@cambridge.co.uk
ETSI Informática, Universidad de Málaga, Málaga, España
Full list of author information is available at the end of the article

**Abstract**

**Keywords:** sample; article; author

## Content

Text and results for this section, as per the individual journal's instructions for authors. Here, we reference the figure **??** and figure **??** but also the table 1.

## Section title

Text for this section...

In this section we examine the growth rate of the mean of $Z_0$, $Z_1$ and $Z_2$. In addition, we examine a common modeling assumption and note the importance of considering the tails of the extinction time $T_x$ in studies of escape dynamics. We will first consider the expected resistant population at $vT_x$ for some $v > 0$, (and temporarily assume $\alpha = 0$)

$$E\big[Z_1(vT_x)\big] = \int_0^{v\wedge 1} Z_0(uT_x)\exp(\lambda_1)\,du.$$

If we assume that sensitive cells follow a deterministic decay $Z_0(t) = xe^{\lambda_0 t}$ and approximate their extinction time as $T_x \approx -\frac{1}{\lambda_0}\log x$, then we can heuristically estimate the expected value as

$$\begin{aligned} &E\big[Z_1(vT_x)\big] \\ &\quad = \frac{\mu}{r}\log x \int_0^{v\wedge 1} x^{1-u}x^{(\lambda_1/r)(v-u)}\,du. \end{aligned} \tag{1}$$

Thus we observe that this expected value is finite for all $v > 0$ (also see [**?**, **?**, **?**, **?**, **?**, **?**]).

**Table 1 Sample table title. This is where the description of the table should go**

| | B1 | B2 | B3 |
|---|---|---|---|
| A1 | 0.1 | 0.2 | 0.3 |
| A2 | ... | .. | . |
| A3 | .. | . | . |

### Sub-heading for section

Text for this sub-heading...

*Sub-sub heading for section*
Text for this sub-sub-heading...

*Sub-sub-sub heading for section* Text for this sub-sub-sub-heading...

# 1 Introducción

Tal y como nos muestra la biología de sístemas, sabemos que un organismo biologico es un sistema integrado e iterrelacionado de genes, proteínas y reacciones bioquimicas, que da lugar a procesos biologicos. Nuestro estudio consiste en ver la relación de los genes viricos pertenecientes al SARS-CoV2 con los genes humanos. Partimos de alrededor de 30 grafos individuales, que muestra la proteina virica y los patógenos unidos a las proteínas humanas. Estos datos al ser descargados nos ofrecen seis ficheros .xslx que muestra el nombre en uniprot, symbol de estas proteinas humanas, junto con algunas descripciones. Nuestros ficheros de interes son Network_table.xlsx y Prey_Lookup_Table.xlsx, analizaremos estos datos y veremos cual nos sera util para el trabajo que queremos llevar a cabo. Nuestro objetivo es tomar cada uno de estos grafos y unirlos entre si mediante interacciones proteinahumana-proteinahumana, para conseguir un unico grafo conexo completo. Para esto usaremos la red del proteoma humano conseguida en string "9606.protein.links.v11.5.txt".

# 2 Materiales y métodos

## 2.1 Obtención de datos en formato String

En este estudio debemos realizar una ampliacion de la red de proteinas humanas unidas a proteinas viricas. Esta ampliacion debe hacerse usando una red de string, que emplea codigo ensamble para nombrar a las proteinas. Nuestros datos están codificados con symbol o Uniprot. Por tanto, debemos hacer un cambio de esta codificacion a ensamble. Para esto hemos empleado tres metodos, y nos quedaremos con aquel que nos permita trabajar con el mayor numero de genes humanos unidos a los genes del covid. Los métodos empleados son: bitR, biomaRt, y utilizar una tabla obtenida de la base de datos de uniprot.

## 2.2 Obtención del grafo de relaciones proteinascovid-proteinasHumanas

Para la realizacion del grafo con una unica componente conexa, donde se forme la red minima del interactoma humano que permita conectar todas las proteínas viricas del SARS-CoV2, hemos usado las funciones all_simple_paths y all_shortest_paths del paquete igraph de R. Debido al gran tamaño de la red y a las miles de combinaciones posibles, obteniamos caminos enormes que se han abordado realizando pequeñas modificaciones que nos permita obtener la componente conexa con un tiempo de ejecucion y uso de memoria razonable.

## 2.3 Obtencion de la modularidad

Para estudiar la modularidad del grafo obtenido hemos utilizado unicamente el paquete iGraph, ya que su combinacion de funciones nos permite obtener justo lo que necesitamos. En este caso, las funciones principales son cluster_walktrap y modularity. Cluster_walktrap encontrara los subgrafos o comunidades en nuestro

grafo a traves de recorridos aleatorios, y modulatiry calculara la modularidad a partir de ellos. A parte, la funcion communities me ha permitido ver claramente los grupos, y posteriormente representarlos con plot.igraph.

### 2.4 Obtencion de la centralidad

La centralidad determina la importancia que puede obtener un nodo dentro de una red. Con el objetivo de estudiar la centralidad de nuestra red, se ha decidido emplear los paquetes **igraph** y **CINNA** principalmente. Mediante el paquete CINNA hemos decidido comprobar la centralidad que se obtiene mediante algunos de los metodos mas comunes como pueden ser el algoritmo **page-rank**, centralidad mediante el grado de cada **nodo**, la centralidad basada en la **cercania** y la centralidad a traves del **betweeness**(capacidad para estar en medio de los paths biologicos importantes). Se ha escogido el algoritmo closeness ya que se ha visto como el más eficiente según la pca.

## 3 Resultados

### 3.1 Obtencion de datos en formato String

Se utilizan los tres metodos mencionados en la seccion de Materiales y Metodos.

#### *3.1.1 bitR*

Para este analisis hemos usado el paquete bitR de Rstudio. Hemos realizado un bitR del proteoma humano para un score maayor de 650. Posteriormente hemos comprobado cuantas de las proteinas humanas de nuestra tabla de union con covid se encuentra en nuestro proteoma obtenido. Posteriormente, hemos comprobado cuantas de ellas estan unidas a genes covid, y cuantos genes covid son.

#### *3.1.2 biomaRt*

Para este analisis hemos utilizado un paquete de Rstudio llamado biomaRt. Este paquete nos permite cambiar la codificacion de uniprot a ensamble. El codigo perteneciente a este analisis se encuentra en la carpeta code y se denomina biomart.R. En este codigo viene detallado con comentarios el proceso que se ha llevado a cabo. Basicamente los pasos seguidos son: - pasar de codigo uniprot a ensamble - comprobar cuantos de estos codigos ensamble se encuentran en el proteoma completo, guardamos en un dataframe los valores ensamble y uniprot. Comprobamos que solo nos quedamos de 332 proteinas humanas unidas a viricas con 107. - Hacemos un bucle y guardamos en un dataframe los valores de uniprot y ensamble aquellos que coinciden con nuestra tabla de entrada. - Hacemos un merge que nos une los dos dataframe, el de entrada con genes covid y humanos, y el obtenido con el cambio de uniprot a ensamble.

#### *3.1.3 Tabla UniProt*

Para este analisis hemos utilizado una tabla obtenida de la base de datos de UniProt, que contiene tres columnas. Un valor de uniprot, uno de ensamble y otro tipo de codigo uniprot para el mismo gen. El codigo perteneciente a este analisis se encuentra en la carpeta code y se denomina tablaObtenidaUniprot.R. Nuevamente en el codigo vienen detallados los pasos seguidos. El proceso llevado a cabo es el siguiente:

- Leemos los ficheros: la tabla de uniprot, la tabla de relaciones entre genes viricos y humanos, y el proteoma de interaccion completo. El fichero con la tabla de uniprot tiene filas que no contienen codigo ensamble para algunos de los genes. Asi que realizaremos un filtrado que elimine estas filas. - Una vez realizado esto, buscaremos cuantos de estos codigos ensamble se encuentran en el proteoma completo. Vemos que solo perdemos cuatro de estos codigos. - Posteriormente, buscaremos cuantos de los codigos uniprot de mi tabla de entrada podemos convertir a ensamble.

### 3.2 Obtencion del grafo de relaciones proteinascovid-proteinasHumanas

Para abordar el problema de la creacion de un grafo conexo, nos hemos enfrentado a tiempos de ejecucion elevados y un gasto de memoria elevada. El codigo denominado AlgoritmoRedCompleta.R muestra como se ha llevado a cabo el proceso de obtencion de red. Una vez obtenido en el apartado anterior la tabla con nuestros genes uniprot en formato string, ya podriamos usar funciones pertenecientes a igraph que buscaran rutas entre una y otra proteina humana dentro del proteoma. Como primer paso se intento obtener todas las rutas posibles entre todas las proteinas humanas unidas al covid, usando la función all simple paths. Nos enfrentamos a una compilacion donde tras 15 horas de ejecucion continua nuestro código no había terminado de definir las rutas entre las proteinas,además de haber generado un archivo de casi 15 gigas que por poco ocupaba toda la memoria de R. Decidimos hacer una pequeña modificaciones y calcular todas las rutas posibles con la misma funcion pero simplemente haciendo las combinaciones de un gen con su siguiente. Nuevamente el tiempo de ejecucion y la memoria ocupada eran inviables. Tambien se investigo el parametro cuttof que permitia establecer un tamaño minimo del camino, pero tampoco conseguiamos un resultado que su tiempo de ejecucion fuera factible. Asi que, tras varios dias de prueba, decidimos usar la función all shortest path, que obtenia la ruta mas rapida de una proteina a otra. Creíamos que esto seria más rapido y obtendriamos los resultados necesarios para poder obtener el grafo. Pero nuevamente, vimos que el tiempo y la memoria que se ocupaba eran enormes. Otra opcion era comparar una proteina con su siguiente, pero esto tras 5 horas y 8 gigas no habia terminado de compilar. Como ultima opción, decidimos hacer una simplificacion. Cogimos 2 genes humanos por cada gen de covid, e hicimos un dataframe. Este dataframe es el que usamos para obtener el grafo completo. Probamos dos opciones, una de ellas utilizaba todas las combinaciones posibles entre estas dos proteinas de cada gen (51 proteinas en total) y la otra utilizaba una proteina con su siguiente. Para optimizacion de tiempo y memoria usamos la segunda de las opciones, obteniendo 100 elementos donde cada una tenia varias listas de uniones de grafos. Como muchas de ellas se repetian usamos unique y conseguimos reducir estas repeticiones. Obteniendo 18012 rutas, que seguiamn teniendo elementos repetidos. A continuación, creamos dos vectores, y hacemos un bucle que vaya recorriendo estas rutas, y me vaya añadiente una proteina en un vector, y la proteina con la que interacciona en el siguiente. Todo esto se integra en un data frame y usamos unique para eliminar las repeticiones. Este dataframe lo convertimos en objeto igraph y mediante el operados de unión conseguimos unir dos objetos igraph, esto unira los dos objetos igraph y ya podremos comprobar si hemos obtenido o no la componete conexa usando la función components. Es importante mencionar que para poder crear las rutas, nuestro grafo de proteoma debe

tener una unica componente conexa, y si tiene mas de una, las proteinas buscadas pertenezcan al mismo. El resultado se comentara en el apartado de discusiones.

### 3.3 Modularidad

Para obtener la modularidad del grafo hemos usado el paquete igraph. El codigo correspondiente se encuentra en un archivo llamado Modularidad y Centralidad.R Lo primero fue obtener los modulos del grafo mediante la función cluster walktrap. Obtuvimos 68 grupos de tamaño diverso, desde 4 hasta 1778. Una vez obtenido esto, se le pasa a la funcion modularity transformándolo a un vector de membership (con la funcion membership) y pasando el grafo original como parámetro. El resultado es una modularidad de 0.935. Despues de esto, representamos algunos valores pertinentes, como una lista de los tamaños de los grupos y Grafo en el que solo se ven los modulos coloreados. Este grafo se hizo con plot.igraph, y nos muestra los 68 modulos por separado. Aparte, vimos la posibilidad de representar un dendograma, pero no aporta ninguna informacion util debido a la cantidad de datos.

## 4 Discusión

### 4.1 Datos a emplear

Tras comparar los resultados de los tres algoritmos anteriores. Vemos que la tabla final que deberemos usar para el analisis se obtiene usando el tercer metodo, donde no perdemos ninguna proteina virica y unicamente perdemos 4 de las humanas. Ya que usando el metodo de biomaRt perdemos 225 genes humanos, perdemos tambien 4 genes viricos pertenecientes al Covid. Y hemos comprobado que usando bitR solo obteniamos 18 de las 332 proteinas humanas y además unicamente dos de las proteinas covid.

### 4.2 Grafo conexo

Despues de todos los intentos por obtener un grafo conexo, mencionados en el apartado de resultados. Hemos conseguido obtener el grafo conexo, puede que no sea el grafo optimo ni contenga todas las rutas, debido a los diferentes problemas vistos, pero hemos conseguido unir mediante rutas de proteinas humanas, las proteinas del Covid. Hemos podido comprobar que el grafo del proteoma total formaba una unica componente conexa, por tanto todas las proteinas se encuentran en el mismo lugar, y pueden usarse las funciones de igraph para conseguir las rutas. Tambien se podria haber utilizado las matrices de adyacencia para obtener rutas, tal y como se vio en clase de teoria, pero preferimos haber optado por el metodo del uso de funciones del paquete igraph. La funcion empleada para ver si el grafo obtenido es conexo es components, donde su parametro no indica el numero de componentes, y su parametro csize indica el numero de vertices diferentes. Por tanto, obtenemos una red formada por 2888 nodos, 9333 enlaces entre ellos y una unica compoente conexa. De la cual podremos analizar su modularidad y analisis funcional.

## 5 Conclusiones

**Abreviaciones**
Indicar lista de abreviaciones mostrando cada acrónimo a que corresponde

**Disponibilidad de datos y materiales**
Debéis indicar aquí un enlace a vuestro repositorio de github.

**Contribución de los autores**
Usando las iniciales que habéis definido al comienzo del documento, debeis indicar la contribución al proyecto en el estilo: J.E : Encargado del análisis de coexpresión con R, escritura de resultados; J.R.S : modelado de red con python y automatizado del código, escritura de métodos; ... OJO: que sea realista con los registros que hay en vuestros repositorios de github.