UNIVERSIDAD NACIONAL DE LA MATANZA

DEPARTAMENTO DE INGENIERÍA E INVESTIGACIONES TECNOLÓGICAS

PROGRAMA PROINCE

CÓDIGO C200

Aplicación de Técnicas de Data Mining para Análisis del Microbioma Humano según Funcionalidades Metabólicas

Director: Santa María, Cristóbal Raúl

Co-director: López, Luis

Integrantes: Juan Otaegui, Ariel Cacho Mendoza, Pablo Martínez, Laura Ávila, Marcelo Soria, Victoria Santa María y Mauro Gilardenghi

Fecha de Inicio: 01/01/2017

Fecha de Finalización: 31/12/2018

Informe final

Sumario

1. Resumen y palabras clave

Los métodos de secuenciación de ADN permiten hoy estudiar comunidades enteras de microorganismos tal como la que constituye el microbioma intestinal humano. Hay un creciente interés médico en este análisis pues las modificaciones que ocurren en la microbiota pueden ser responsables de la disbiosis asociada con enfermedades como el cáncer colorectal sobre el cual se focaliza esta línea de investigación. El proceso bioinformático de las muestras desde que salen las lecturas del secuenciador hasta que pueden ser explotadas como datos requiere una sistematización y en lo posible una automatización a la que este trabajo colabora estableciendo una "pipeline" de procesos ligados a través de programas confeccionados al efecto. También se aplican distintas técnicas de explotación de datos para buscar asociaciones y patrones que vinculen la clasificación taxonómica y las vías metabólicas presentes con la condición clínica de los pacientes analizados. Se relatan aquí también los avances realizados para la obtención de muestras de pacientes autóctonos, las vinculaciones alcanzadas con instituciones médicas y los estudios de posgrado generados a partir de los contenidos del proyecto.

Palabras Clave: Cáncer-Microbioma-Secuenciación-Explotación de Datos

2. Memoria descriptiva

INTRODUCCIÓN

Las tecnologías de nueva generación para la secuenciación de ADN permiten tratar cada vez mayor cantidad de muestras a menores costos. Esto ha potenciado notablemente las posibilidades de los estudios metagenómicos, que involucran el conocimiento simultáneo de los genes de todos los individuos que forman una comunidad, extendiendo sus alcances al análisis de la composición microbiana de suelos, aguas y al microbioma humano. Éste no es otra cosa que la comunidad de microorganismos presentes en el cuerpo humano que contiene diez veces más microorganismos que células propias. Se han presentado entonces probabilidades ciertas de evaluar la interacción entre esta microbiota y el organismo alojante que resulta clave en el mantenimiento de la inmunidad y la protección contra agentes patógenos externos al organismo humano. La composición del microbioma varía entre las personas según el estilo de vida, la dieta y su genotipo pero es estable dentro de una misma persona. Si se producen modificaciones de tipo permanente esto conlleva una disbiosis que es la alteración de la influencia de la comunidad en los procesos metabólicos de la persona y que se asocia con enfermedades tales como la inflamación intestinal, el asma o los desórdenes mentales. En particular la disbiosis está implicada en la carcinogénesis al ser iniciadora de los procesos inflamatorios y su presencia da señal de inmunodepresión [1]

Algunos argumentos indirectos sugieren el rol potencial de la microbiota intestinal en la carcinogénesis colorectal. El cáncer colorectal es básicamente una enfermedad genética pero el microbioma alojado por el paciente puede explicar la interacción entre los genes del paciente y el entorno de microorganismos presentes que se manifiesta tanto en su diversidad y riqueza taxonómica cuanto en las vías metabólicas que tienen lugar. Frecuentemente parecen asociados el cáncer colorectal y las variaciones de las frecuencias con que algunas especies bacterianas se encuentran en el microbioma[2] Esta asociación

no es clara aún para determinar si la variación del microbioma es una causa o un efecto del cáncer. Incluso recientemente se ha sugerido que el microbioma puede jugar el rol de control sobre la enfermedad. En todo caso existe una perspectiva interesante en los estudios metagenómicos pues no solo permiten la determinación taxonómica de la comunidad microbiana a través de la utilización de genes marcadores sino que también, al utilizar la información de todos las secuencias obtenidas del microbioma (WGS), pueden establecer las vías metabólicas que potencialmente sigan los procesos celulares en el paciente. Esto ha motivado un profundo interés en la comunidad médica que ha buscado así relacionarse con los campos de la biología, la computación, la estadística y específicamente con la bioinformática para avanzar en la comprensión y eventualmente en el diagnóstico y pronóstico de enfermedades.

Al respecto de lo hasta aquí expuesto debe señalarse que no solo los avances en la tecnología de secuenciación han permitido estos estudios cada vez más amplios y profundos sino que también se han producido importantes desarrollos de algoritmos cuya rapidez y precisión de cálculo ha sido creciente a la vez que ha permitido una mayor cantidad de procesos de explotación de datos tanto en aspectos estadísticos descriptivos cuanto en técnicas de aprendizaje automático supervisado y no supervisado. En lo referido al microbioma humano se ha hecho evidente la necesidad de contar con un esquema seriado de procesos computacionales a aplicar desde que las secuencias salen del secuenciador hasta que resultan transformadas en información útil para la investigación clínica. Esto involucra la confección de software de filtrado de las secuencias, de evaluación de contaminación del conjunto con secuencias humanas, de ensamblado de secuencias, de anotación de las mismas según sus niveles taxonómicos, de identificación de vías metabólicas presentes, de agrupamiento en conglomerados o clusters según taxonomía o metabolismo, de aprendizaje sobre conjuntos de entrenamiento y testeo para clasificar microbiomas según los mismos principios. Una gran mayoría de estos desarrollos se realizan en forma de software libre para que puedan ser utilizados y testeados por investigadores a nivel global y se encuentran muchas veces disponibles en repositorios internacionales que también contienen los datos de las distintas experiencias realizadas.

El trabajo que aquí se informa consistió entonces en la construcción de una pipeline que permitiese llegar de las secuencias hasta el proceso bioinformático de aprendizaje automático para clasificación de casos. Se tomó un conjunto de secuencias de microbiomas correspondiente a pacientes con antecedentes de cáncer de colon extraído del repositorio de NCBI [3] y se fueron realizando en línea los distintos procesos que pudieran finalmente como resultado revelar aspectos clínicos de interés. En ese camino muchas veces fue necesario estudiar con detenimiento aspectos matemáticos de los algoritmos y otras hubo que desarrollar programas en lenguaje R o C para lograr unir una etapa de proceso con otra. Se procuró además constatar los propios resultados con los obtenidos en estudios similares. También se comenzó a trabajar en la obtención de una muestra de pacientes locales a efecto de caracterizar en el futuro similitudes y diferencias clínicas al aplicarle la línea de procesos ahora establecida. En tal sentido se abrió a futuro la colaboración con instituciones médicas y con otras universidades a través de convenios que se encuentran en trámite. Por otra parte, en el marco de la investigación algunos integrantes del equipo realizan estudios de posgrado.

MATERIALES Y MÉTODOS

Se inició el análisis de las muestras correspondientes a secuenciación de ADN total de un estudio depositado en la base de datos BIOPROJECT del NCBI con el código PRJNA397450. Este estudio consistió en la extracción y secuenciación de ADN microbiano total y del gen que codifica para el ARN ribosomal de 16S (16S rRNA), a partir de muestras de hisopados rectales, biopsias por endoscopia y materia fecal. Seleccionados dentro de un programa de prevención del cáncer colorectal según reglas estándar tanto estadísticas como de protocolo médico, los pacientes elegidos fueron adultos de entre 40 y 85 años, de buena salud, con antecedentes de pólipos colorectales. Se tomaron muestras en dos momentos diferentes de tiempo espaciados en tres meses. Se procuró además que en la muestra total hubiera un 50% de pacientes con recurrencia en los pólipos y un 50% que no. Finalmente 60 individuos fueron seleccionados a lo largo de dos años sobre un total de 150 participantes. Más características de la muestra tomada son explicitadas en [4].

Los datos de secuenciación del estudio son entonces públicos y al momento de comenzar el trabajo no tenían ninguna publicación asociada. Esto no fue un inconveniente, pues lo necesario era que los datos permitieran poner a punto, y en lo posible automatizar, las operaciones bioinformáticas comunes sobre datos de este tipo: la selección de los softwares más adecuados para cada una de dichas operaciones, el análisis de calidad de las secuencias, el diseño de una metodología de limpieza de las secuencias, su posterior validación, el ensamblado de los metagenomas y finalmente la implementación de la “pipeline” que permitiera integrar todos los pasos y automatizar la mayor cantidad. El estudio cuenta con 143 muestras que se distribuyen de la siguiente manera: 16 de endoscopias, 99 de materia fecal y 28 de hisopados rectales. La secuenciación de ADN total se realizó con tecnología Illumina con una estrategia de “paired-ends” de 300 nucleótidos, por lo que cada muestra está compuesta por dos archivos de secuencias. Esto significa que para cada fragmento de ADN analizado, se secuencian 300 nucleótidos desde cada extremo.

Todos los pasos que se detallan a continuación se realizaron en computadoras corriendo el sistema operativo Linux, distribución Ubuntu 16.04. Se procedió a la descarga de los 286 archivos utilizando la herramienta SRATOOLKIT que distribuye el NCBI y se eliminaron dos muestras (4 archivos) que contaban con muy pocas secuencias. Las muestras restantes tenían entre, aproximadamente, 41600 y 521000 bases.

Se realizó el control de calidad de estas secuencias con el software FastQC [5] y se determinó que casi todas las secuencias tenían restos de dos de los adaptadores que usa Illumina para la secuenciación, una en las secuencias F (“forward”) y otro en las secuencias R (“reverse”) de cada “paired end”.Además se determinó que las frecuencias de cada base en los primeros 15 nucleótidos de las secuencias presentaban un nivel de variabilidad muy alto, que no era compatible con lo que se observaba más adelante y debido, posiblemente, a algún artefacto de la secuenciación que generaba “ruido”. También la calidad promedio de las secuencias caía por debajo del valor umbral que se fijó en 25 a partir de una posición que variaba para cada secuencia, pero que en general se ubicaba después de la posición 240. Para determinar el tipo exacto de secuencia contaminante y para tener una información más precisa del lugar en que ocurría utilizamos el software Scythe [6]. El proceso de

limpieza se realizó con el programa Cutadapt [7] que permite realizar limpieza de adaptadores, cortes por caída en los valores de calidad, eliminación por largo mínimos, cortes en posiciones arbitrarias, etc. En primer término, se realizaron una serie de pruebas preliminares para determinar las opciones específicas de limpieza y los valores óptimos de los diferentes parámetros del programa. El proceso definitivo se efectuó en dos pasos. En el primero se eliminaron las secuencias contaminantes, se eliminó la parte 3'de las secuencias que presentaran una caída en su calidad por debajo del valor umbral 25 mencionado antes y si alguna de las secuencias de un par "paired-end" después de estos cortes resultó con una longitud menor a 50 bases se procedió a eliminar el par completo. En el segundo paso se eliminaron los primeros 15 nucleótidos del extremo 5' y se volvieron a filtrar los pares para eliminar aquellos con al menos un miembro de longitud menor a 50 bases. Después de este proceso de limpieza se volvió a revisar la calidad de las secuencias con FastQC, que mostró resultados satisfactorios.

Con el objetivo de obtener la caracterización taxonómica y funcional de las muestras se llevaron a cabo luego distintos procesos. Por un lado, se trabajó directamente con los reads. que son las lecturas obtenidas de la secuenciación, y por otro se enfocó la tarea hacia la obtención de los contigs (reads ensamblados). Se procuró establecer adecuadamente la cadena de procesos [8] en este último caso. La pipeline utilizada fue la siguiente:

1. Ensamblado.

El proceso de ensamblado consiste en unir los reads para obtener secuencias más largas, contigs, que se corresponden con las secuencias de ADN antes de haber sido cortadas para su secuenciación. Al respecto se estudiaron en detalle los aspectos matemáticos y se realizaron pruebas con los programas IDBA-UD [9] y Megahit [10]. De estos estudios y pruebas surgió el review "Treatment of Massive Metagenomic Data with Graphs" que fue publicado en las proceedings de las VI Jornadas de Cloud Computing & Big Data y que se adjunta en Anexo III. Finalmente se decidió utilizar Megahit, con una longitud mínima de 400 bases por contig. [11]

2. Identificación y eliminación de secuencias humanas.

Una vez obtenidos los contigs, se eliminaron aquellos correspondientes a la contaminación humana. Para este proceso se alinearon los contigs contra el genoma humano de referencia GRCh38, utilizando Blastn. Los contigs que alinearon contra el genoma humano se eliminaron del juego de datos utilizando un script R desarrollado para tal fin.

3. Anotación taxonómica.

La anotación taxonómica de los contigs no humanos se realizó utilizando Kaiju junto

con su propia base de datos. [12] y [13]

4. Anotación funcional**.**

La anotación funcional inicial de los contigs no humanos se realizó con Prokka. Luego se agregaron anotaciones KEGG usando el servicio KAAS [14]

5. Proceso de Recuento:

A continuación, se usó el software Bowtie2 [15] para generar los índices de los contigs y luego se alinearon los reads contra esos contigs. De esta forma se obtuvieron los mapeos de los reads a la referencia en archivos sam. Luego, utilizando samtools se los convirtió a bam y se indexaron estos mapas. Con samtools se obtuvo la cantidad de reads que mapearon abriendo la perspectiva a estudios del microbioma basados directamente en los reads aunque se prefirió por ahora analizarlo desde los contigs pues se poseía al respecto mayor información.

6. Consolidación de los resultados

Para llevar a cabo los procesos de análisis estadístico y de minería de datos se construyeron una serie de scripts en R que consolidan en una única tabla la información de anotación funcional para cada gen predicho con Prokka y KAAS, de anotación taxónomica de KAIJU a nivel de contigs y variables adicionales como el largo de cada contig, el largo de cada gen predicho. La forma en que allí se disponen los datos puede verse en las líneas de ejemplo de la Tabla 1

Tabla 1. Ejemplo de Información Consolidada

sample_ID	contig_ID	ID	reino	phylum	clase	orden	familia	genero	KO
SRR5907479	k141_3	GKLBDHMD_00001	Bacteria	Actinobacteria	Actinobacteria	Actinopolysporales	Actinopolysporaceae	Actinopolyspora	NA
SRR5907479	k141_4	GKLBDHMD_00002	Bacteria	Firmicutes	Clostridia	Clostridiales	Lachnospiraceae	NA	K03282
SRR5907479	k141_5	NA	Bacteria	Firmicutes	NA	NA	NA	NA	NA
SRR5907479	k141_7	GKLBDHMD_00003	NA	NA	NA	NA	NA	NA	K01424
SRR5907479	k141_8	NA	NA	NA	NA	NA	NA	NA	NA
SRR5907479	k141_9	NA	Bacteria	NA	NA	NA	NA	NA	NA
SRR5907479	k141_10	GKLBDHMD_00004	Bacteria	Proteobacteria	Alphaproteobacteria	Rhodospirillales	Rhodospirillaceae	Rhodospirillum	K02120
SRR5907479	k141_11	GKLBDHMD_00005	Bacteria	Firmicutes	Clostridia	Clostridiales	Clostridiaceae	Clostridium	NA
SRR5907479	k141_12	NA	Bacteria	Proteobacteria	Alphaproteobacteria	Caulobacterales	Caulobacteraceae	Asticcacaulis	NA

En la Tabla 1 la columna primera identifica la muestra, la segunda el contig y luego, de reino a género, se anotan los distintos niveles taxonómicos de la rama del árbol filogenético a la que el contig pertenece. La columna KO contiene las anotaciones de las posibles funcionalidades metabólicas de la secuencia según la Kyoto Encyclopedia of Genes and Genomes (KEGG) [16] lo que permitirá estudiar cada microbioma y al conjunto de todos los que conforman la muestra desde el punto de vista de las funcionalidades metabólicas que puedan estar presentes.

7. Disposición de los datos para su explotación en tablas de frecuencias

El trabajo realizado hasta aquí lleva a las puertas del procesamiento estadístico inteligente. Sin embargo, resta aún la importante tarea de disponer los datos en esquemas y formatos que puedan ser leídos y procesados por los paquetes estadísticos y de aprendizaje automático. Tal tarea se resolvió desarrollando dos programas en lenguaje C, cuyas codificaciones se adjuntan en el Información Adicional, que devolvieron las frecuencias absolutas con que los distintos taxones y anotaciones KO se presentaban en cada microbioma. De esta forma se eligió trabajar con las tablas Microbiomas.Phylum y Microbiomas.KO pues la primera da información sobre los Phylum, entre ellos por ejemplo las fusobacterias cuya especie Fusobacteruim Nucleatum resulta prevalente en el carcinoma humano colorectal [2]. Así mismo las anotaciones KO permitirían clasificar los microbiomas, es decir los pacientes, según las funciones metabólicas que pudieran estar teniendo lugar a nivel celular identificando con ello condiciones clínicas de interés. Las

Tablas 2 y 3 muestran un fragmento a título de ejemplo de la forma en que se dispone la información sobre frecuencias estadísticas para los Phylum y las anotaciones KO respectivamente.

Tabla 2. Ejemplo de Frecuencias Estadísticas para Phylum

Microbioma	Acidobacteria	Actinobacteria	Aquificae	Armatimonadetes	Bacteroidetes
PhSRR5907479	0	108	0	1	49
PhSRR5907480	29	935	10	0	1046
PhSRR5907481	16	394	1	0	570
PhSRR5907482	10	507	1	2	778
PhSRR5907483	34	770	4	4	627
PhSRR5907484	1	102	0	0	55
PhSRR5907485	41	1443	9	4	1926
PhSRR5907486	16	779	4	3	664
PhSRR5907487	13	197	1	1	139
PhSRR5907488	3	126	0	0	84
PhSRR5907490	7	313	1	4	309
PhSRR5907491	2	118	0	0	56
PhSRR5907492	0	3	0	0	1

En la primera columna de la Tabla 2 se ven los códigos relativos a cada microbioma. Desde la segunda columna en adelante hasta la número 55 corresponden a los nombres de los distintos Phylum que fueron hallados entre las muestras. Cada número en el interior de la tabla corresponde a la frecuencia absoluta de un taxón Phylum en un dado microbioma. Lo propio ocurre con la Tabla 3 donde las columnas son las anotaciones KO halladas que en este caso fueron 4000.

Tabla 3. Ejemplo de Frecuencias Estadísticas para Anotaciones KO

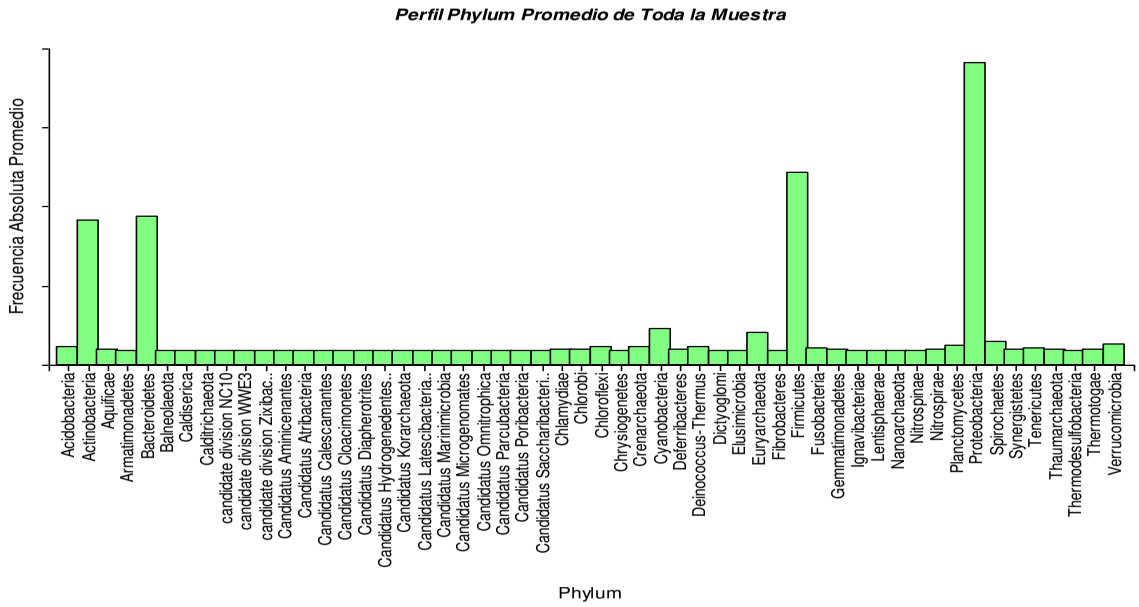
	K00002	K00003	K00005	K00008	K00009	K00010	K00011	K00012	K00013	K00014	K00015
SRR5907479	0	0	0	0	0	0	0	0	0	0	0
SRR5907480	0	0	0	0	0	0	0	1	0	0	0
SRR5907481	0	0	0	0	0	0	0	0	0	1	0
SRR5907482	0	0	0	0	0	0	0	0	0	1	0
SRR5907483	0	0	1	0	0	0	0	1	0	1	0
SRR5907484	0	0	0	0	0	0	0	0	0	0	0
SRR5907485	0	0	0	0	0	0	0	3	2	1	0
SRR5907486	0	0	0	0	0	0	1	0	0	1	0

8. Explotación de datos

Para procesar los datos de las tablas Microbiomas.Phylum y Microbiomas.KO se utilizaron los softwares Weka [17] e Infostat [18]. Los procedimientos aplicados sobre los datos a nivel taxonómico Phylum fueron los siguientes:

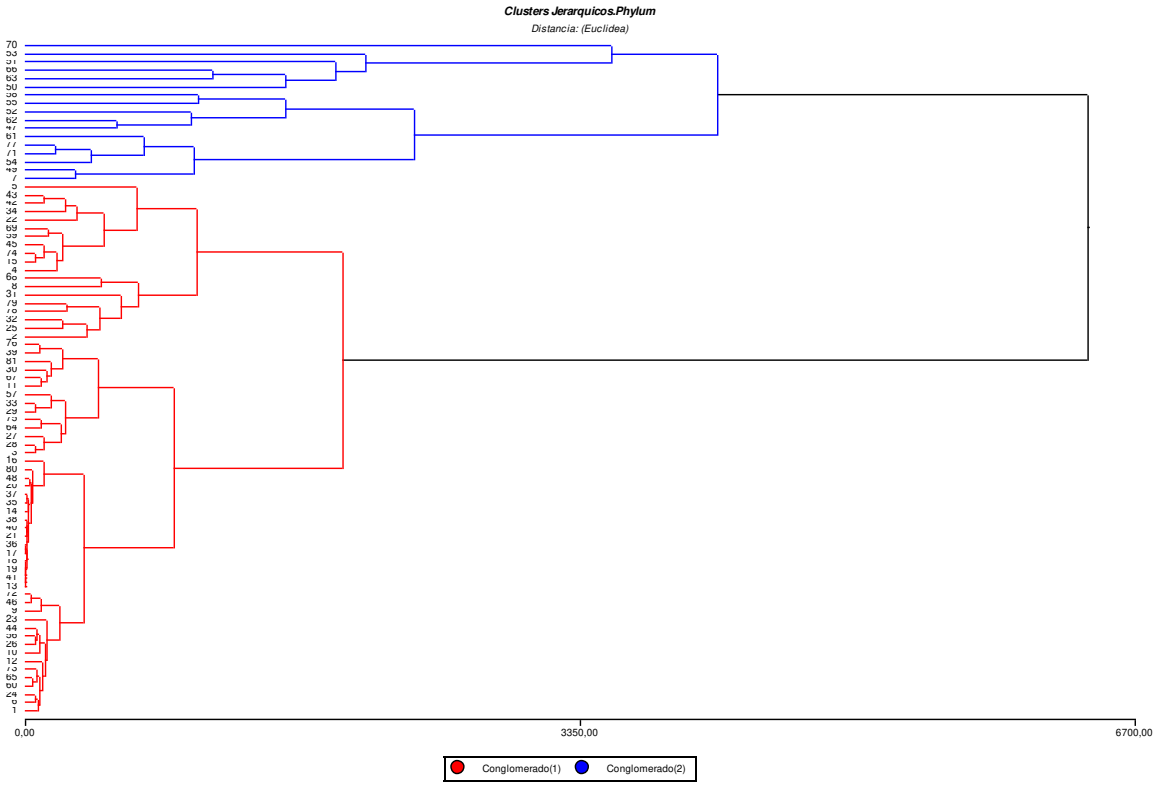
- Con el objetivo de tener una primera impresión sobre la diversidad y riqueza de los distintos Phylum en la muestra total se calcularon medidas estadísticas de resumen determinando para cada Phylum su media y suma total. La Figura 1 muestra el promedio de la frecuencia absoluta de cada Phylum constituyendo el perfil promedio de distribución.

Figura 1. Perfil Phylum Promedio para Toda la Muestra



- A continuación, se procuró determinar alguna forma de agrupamiento que mostrara posibles diferencias que analizadas luego clínicamente tuvieran relevancia. Se analizaron distintas metodologías para formar conglomerados. Para establecer los clusters en forma jerárquica se utilizó la distancia euclídea con encadenamiento promedio eligiéndose la formación de 2 agrupamientos que se volcaron en el dendrograma de la Figura 2.

Figura 2. Clusters Jerárquicos Phylum



Por otra parte, la aplicación del método K-means dio también por elección automática óptima dos clusters integrados exactamente por los mismos pacientes que los de los agrupamientos jerárquicos. Se adosó a la tabla Microbioma.Phylum la variable Conglomerado que identifica el grupo al cual pertenece cada microbioma.

- En la búsqueda de aquellas diferencias mencionadas entre grupos se calcularon para cada uno las medidas de resumen media y suma total. Las Figuras 3 y 4 muestran los respectivos perfiles promedio.

Figura 3. Perfil Phylum de Promedios- Conglomerado 1

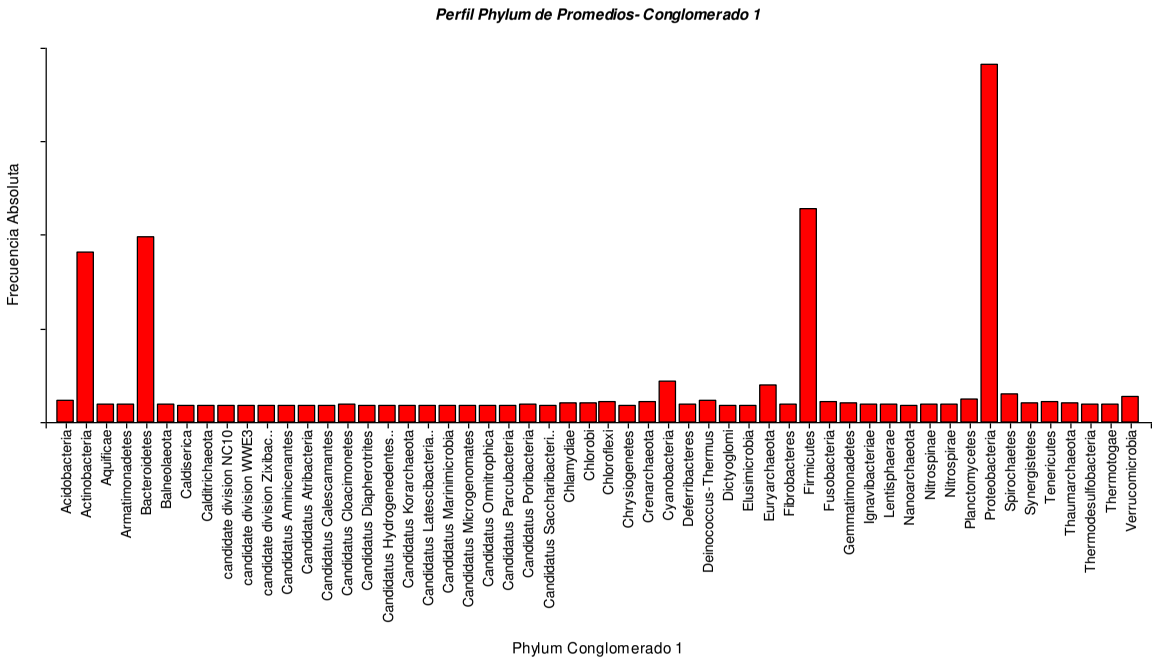
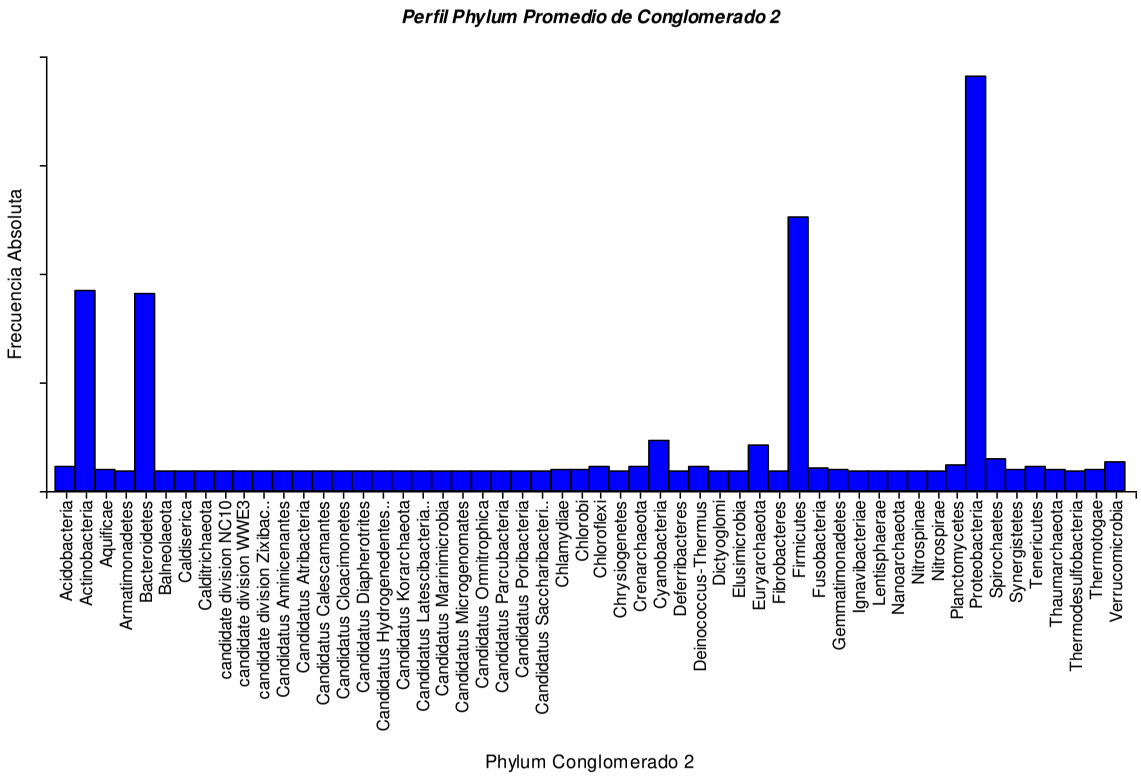
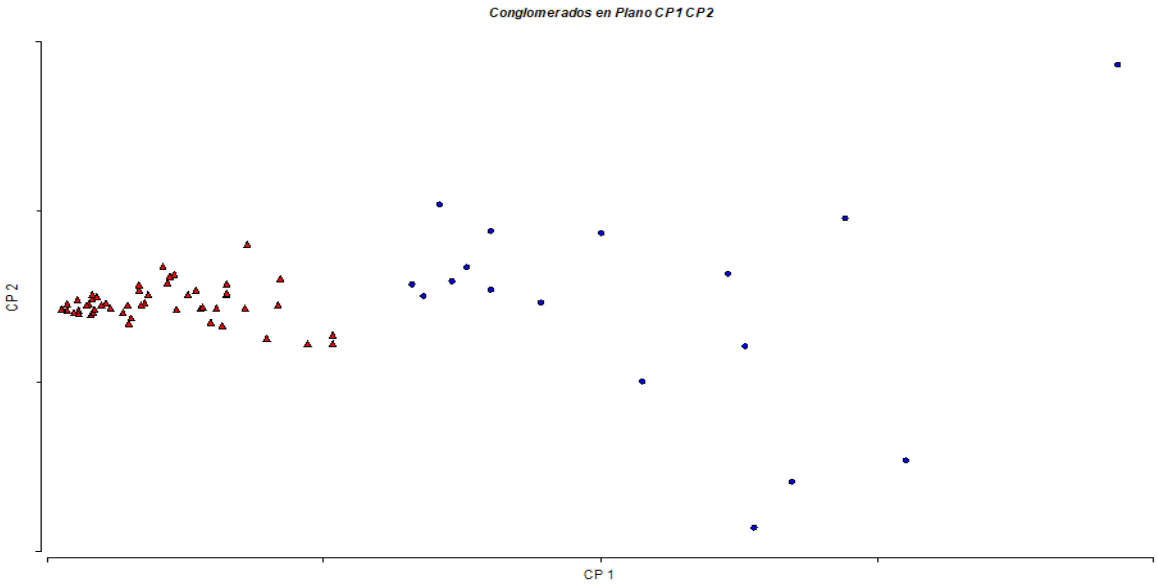


Figura 4. Perfil Phylum Promedio de Conglomerado 2



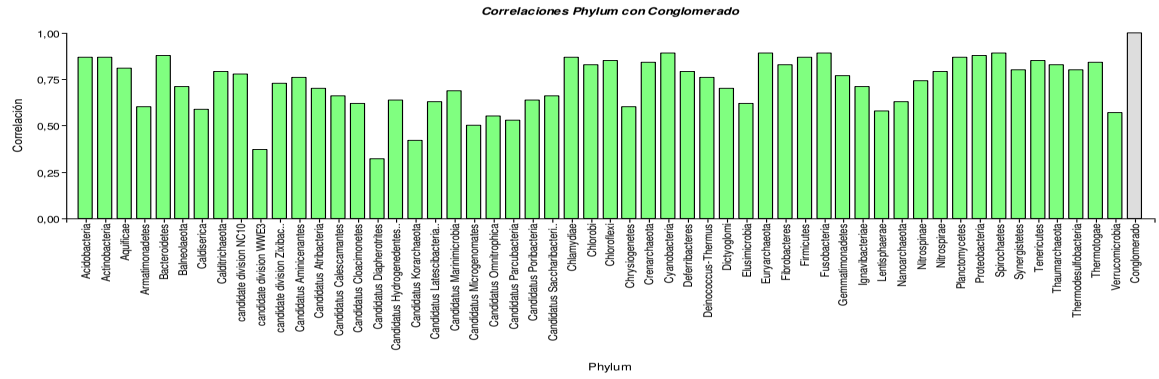
- Luego, a efecto de reducir la cantidad de variables buscando explicar al menos la parte substancial de la clasificación por conglomerado, se aplicó sobre todo el conjunto de microbiomas, el método de componentes principales. Para todo el conjunto de microbiomas se seleccionaron las dos primeras variables: CP1, que explica el 71% del comportamiento de las variables Phylum, y CP2 que explica el 4%. Se graficó la variable Conglomerado en el plano CP1-CP2 obteniéndose la Figura 5.

Figura 5. Conglomerados en Plano CP1-CP2



- Para determinar aquellos Phylum mas relacionados estadísticamente entre si y a su vez los más relacionados con la variable de clasificación conglomerado, se estudió la correlación lineal entre variables. En primer término, se graficó la correlación entre cada una de las variables Phylum con la variable Conglomerado como se ve en la Figura 6.

Figura 6. Correlaciones Phylum con Conglomerado



Luego se calculó la correlación de las variables Phylum con las componentes principales CP1 y CP2 lo que se muestra en las Figuras 7 y 8.

Figura 7. Correlación Phylum con CP1

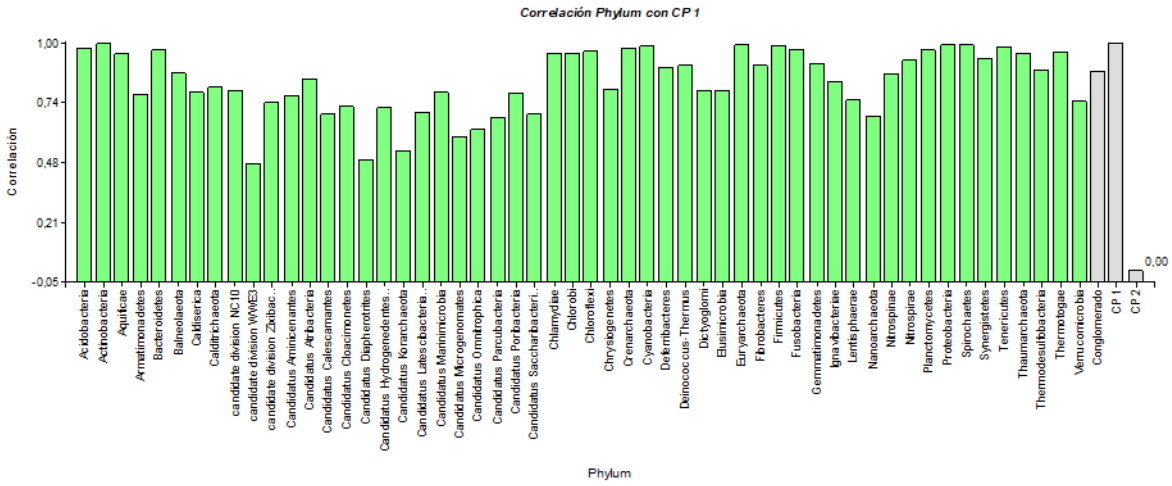
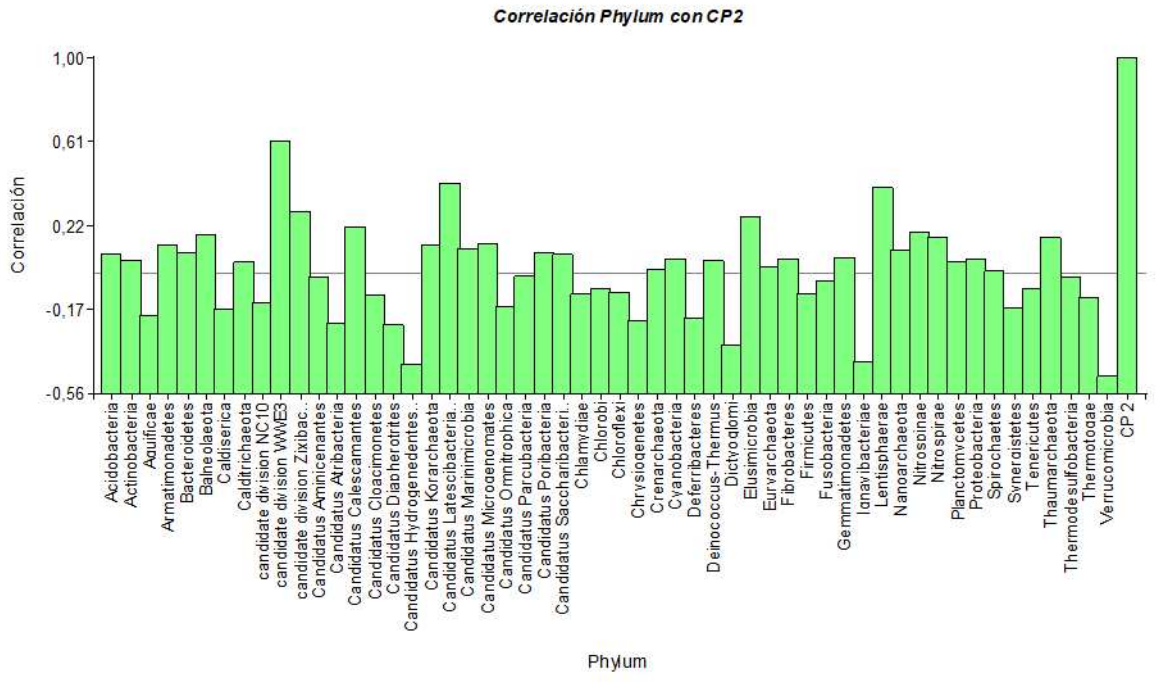


Figura 8. Correlación Phylum con CP2



En la búsqueda de alternativas que revelaran las variables claves asociadas a la clasificación dada por el número de conglomerado, se analizó además la relevancia de cada variable Phylum en la determinación del conglomerado por vía de un algoritmo de selección de atributos que evalúa el valor de cada subconjunto de atributos considerando la habilidad predictiva en cada caso junto al grado de redundancia entre las variables [19]. Los resultados se ven en la Tabla 4.

Tabla 4. Selección de Atributos

=== Attribute Selection on all input data ===

Selected attributes: 8,9,11,12,14,25,26,37,39,43

Calditrichaeota

candidate division NC10

candidate division Zixibac..

Candidatus Aminicenantes

Candidatus Calescamantes

Candidatus Saccharibacteri..

Chlamydiae

Fibrobacteres

Fusobacteria

Nanoarchaeota

- A continuación se consideraron por separado los conglomerados y se calcularon las componentes principales con el objetivo de estudiar en cada caso la representatividad que pudieran ostentar sobre las variables Phylum. Se estableció que para el Conglomerado 1 el eje CP1 explica el 49% del comportamiento de la variable Phylum mientras que el CP2 agrega un 5% más. A su vez en el Conglomerado 2 la componente principal CP1 representa el 48% del comportamiento de Phylum pero el CP2 alcanza un 11%. Además, para cada conglomerado se calculó la correlación de Phylum con sus componentes principales lo que se grafica en las Figuras 9,10,11 y 12 respectivamente.

Figura 9. Correlación Phylum con CP1-Conglomerado 1

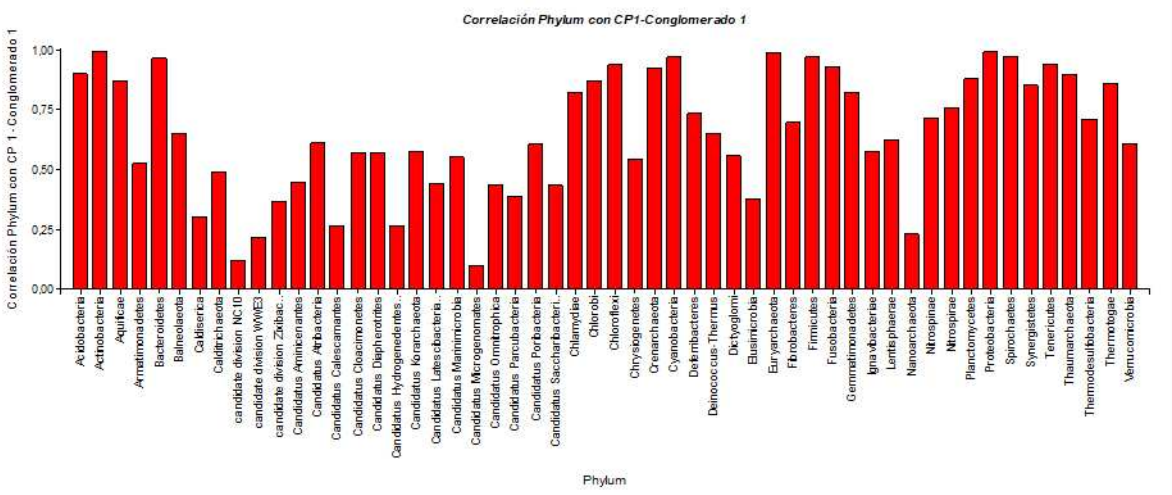


Figura 10. Correlación Phylum con CP2-Conglomerado 1

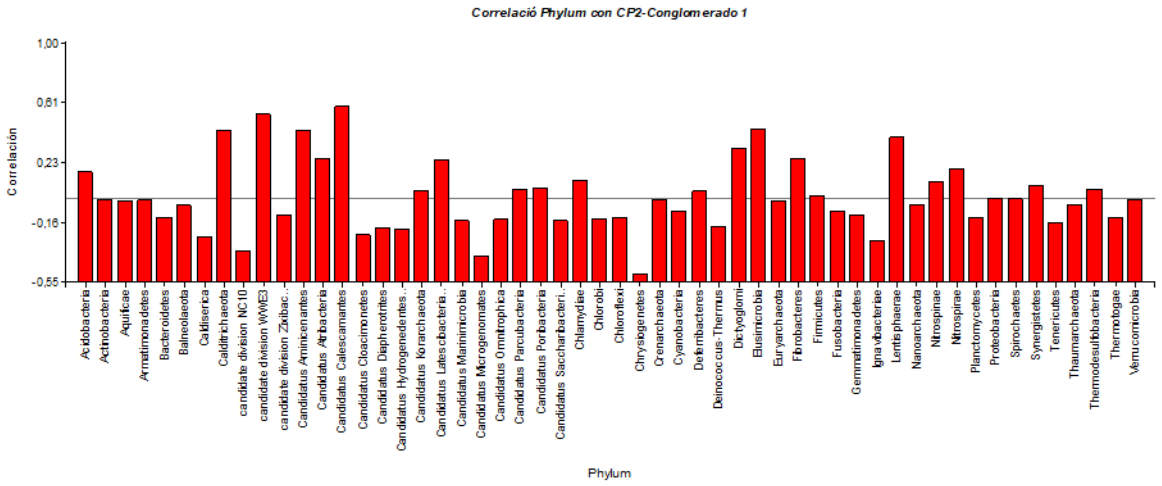


Figura 11. Correlación Phylum con CP1-Conglomerado 2

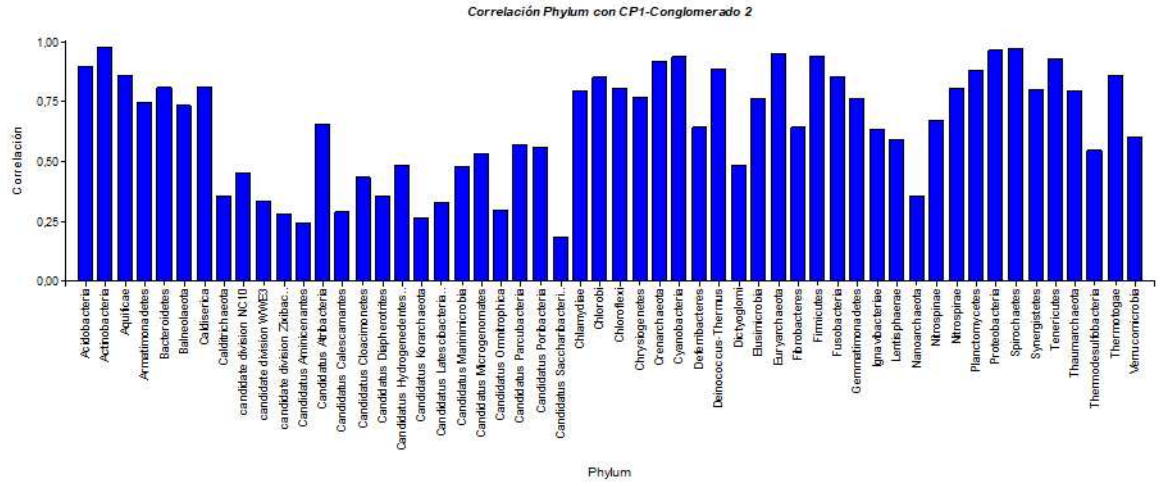
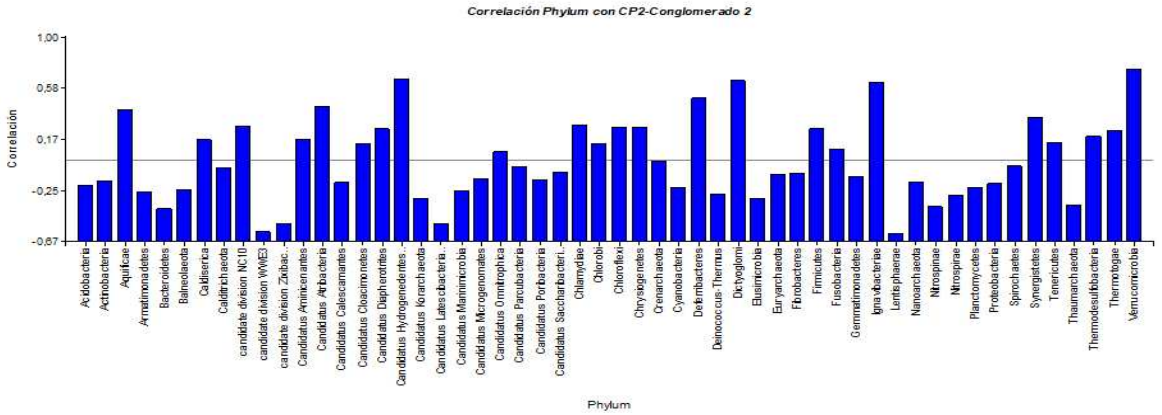


Figura 12. Correlación Phylum con CP2-Conglomerado 2



- Finalmente se utilizó la variable Conglomerado a efecto de entrenar y testear un árbol de decisión para predecir la clasificación de un paciente aún no catalogado en ninguno de los conglomerados. Primero se realizó el entrenamiento de un árbol J48 mediante un subconjunto formado por 38 microbiomas. Luego se llevó a cabo el testeo con un subconjunto distinto de 43 microbiomas. La Tabla 5 muestra el desempeño del árbol en el entrenamiento y la Tabla 6 informa los resultados del testeo.

Tabla 5. Entrenamiento Árbol J48 Phylum

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     Entrenamiento-
Instances:    38
Attributes:   55

Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
=== Summary ===
Correctly Classified Instances          35               92.1053 %
Incorrectly Classified Instances         3                7.8947 %
=== Detailed Accuracy By Class ===

TP Rate FP Rate ROC Area  Class
0,933    0,125   0,904     a
0,875    0,067   0,904     b

=== Confusion Matrix ===

  a  b   <-- classified as
 28  2 |  a = a
  1  7 |  b = b
```

Tabla 6. Testeo Árbol J48 Phylum

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     Testeo-weka.filters.unsupervised.attribute.Remove-R1,56
Instances:    43
Attributes:   55

Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------
```

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          39               90.6977 %
Incorrectly Classified Instances         4                9.3023 %

=== Detailed Accuracy By Class ===

TP Rate  FP Rate  ROC Area  Class
0,889    0,088    0,900     b
0,912    0,111    0,900     a
=== Confusion Matrix ===

  a  b   <-- classified as
  8  1 |  a = b
  3 31 |  b = a
```

De acuerdo a la evaluación de Stantikov et al [20] el algoritmo Random Forest [21] resulta el de óptimo desempeño en datos de microbiomas por lo que se procedió a construir el ensamble de árboles respectivo. En este caso los resultados de entrenamiento y testeo pueden verse en las Tablas 7 y 8

Tabla 7. Entrenamiento Ensamble Random Forest Phylum

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest
Relation:     Entrenamiento-
Instances:    38
Attributes:   55
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          38              100      %
Incorrectly Classified Instances         0                0      %

Total Number of Instances               38

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  ROC Area  Class
                1,000    0,000    1,000     a
                1,000    0,000    1,000     b
```

```
=== Confusion Matrix ===

  a  b   <-- classified as
 30  0 |  a = a
  0  8 |  b = b
```

Tabla 8. Testeo Ensamble Random Forest Phylum

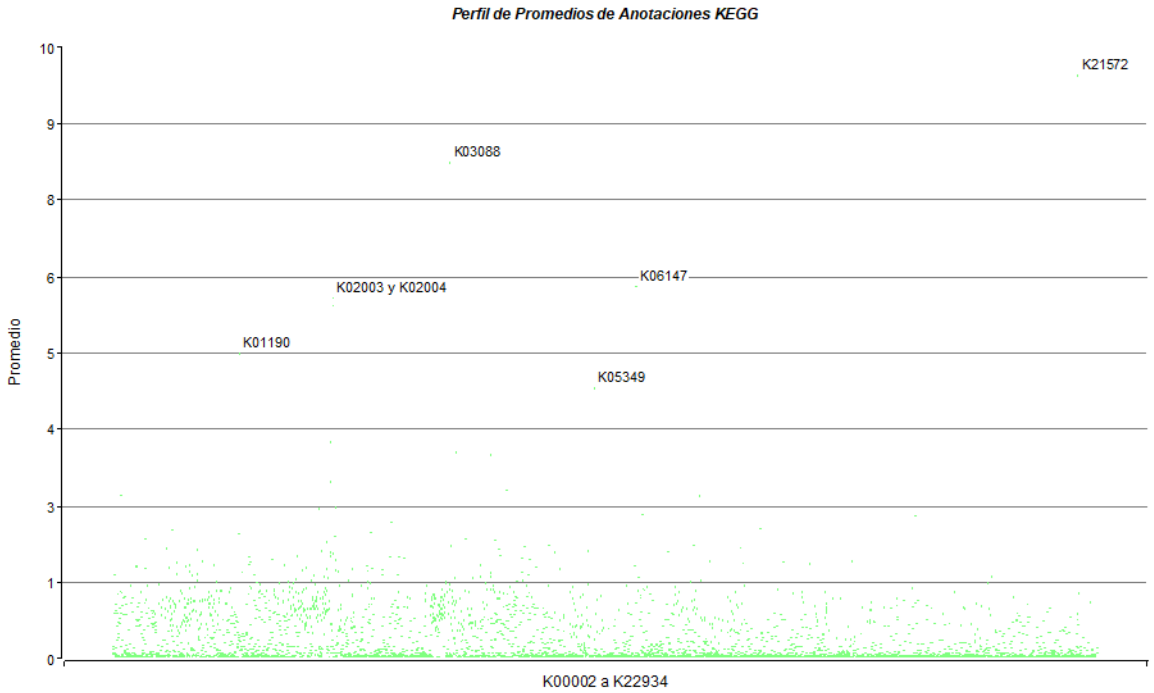
```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest
Relation:     Testeo-weka.filters.unsupervised.attribute.Remove-R1,56
Instances:    43
Attributes:   55
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

weka.classifiers.trees.RandomTree -K 0 -M 1.0 -V 0.001 -S 1 -do-not-
check-capabilities

Time taken to build model: 0.03 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          43               100      %
Incorrectly Classified Instances         0                 0      %

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  ROC Area  Class
                 1,000    0,000       1,000     b
                 1,000    0,000       1,000     a


=== Confusion Matrix ===

  a  b   <-- classified as
  9  0 |  a = b
  0 34 |  b = a
```

Sobre las anotaciones KO asociadas con funcionalidades metabólicas se realizaron los procesos que se detallan a continuación:

- Se calcularon medidas estadísticas de resumen determinando para cada anotación KO su suma, máximo y promedio sobre todo el conjunto de microbiomas. La

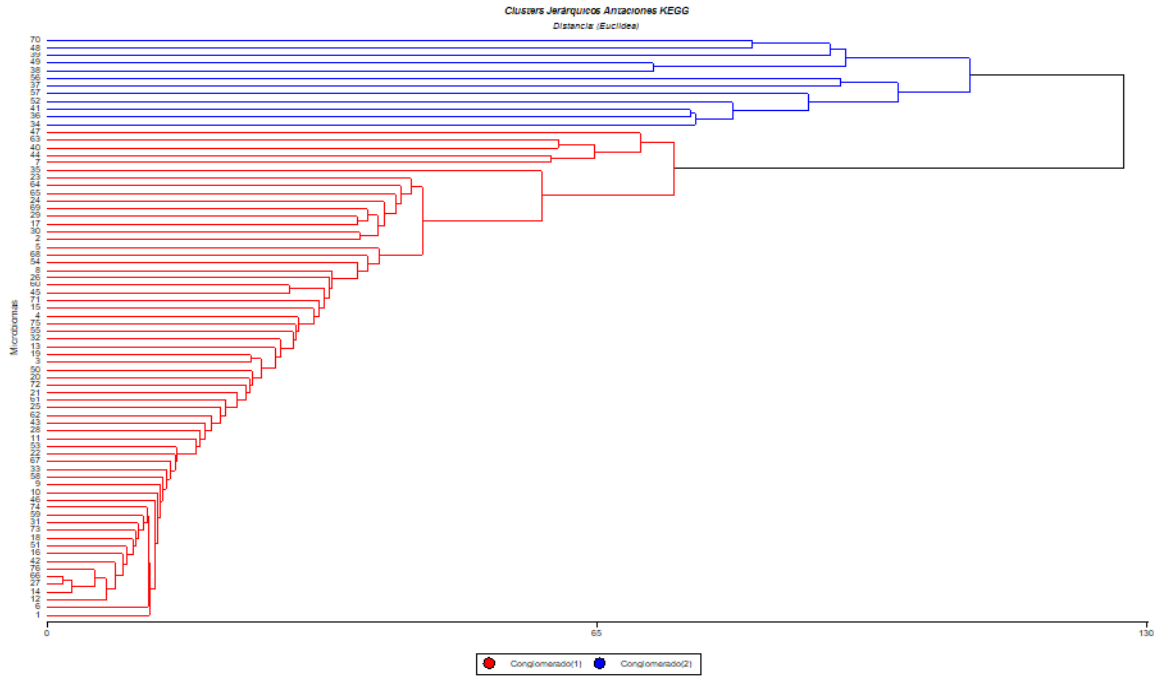
información sobre los promedios se ve en la Figura 13. Allí se han señalados las categorías KO con promedio mayor.

Figura 13. Perfil de Promedios de Anotaciones KEGG



- Se formaron conglomerados por el método jerárquico incorporándose la nueva variable Conglomerado con valores 1 o 2 a la tabla de Microbiomas.KO. La Figura 14 muestra el dendrograma correspondiente.

Figura 14. Clusters Jerárquicos Anotaciones KEGG



Los promedios de frecuencias de aparición de cada anotación KO en los respectivos conglomerados se ven en las Figuras 15 y 16.

Figura 15. Perfil de Promedios de Anotaciones Kegg-Conglomerado 1

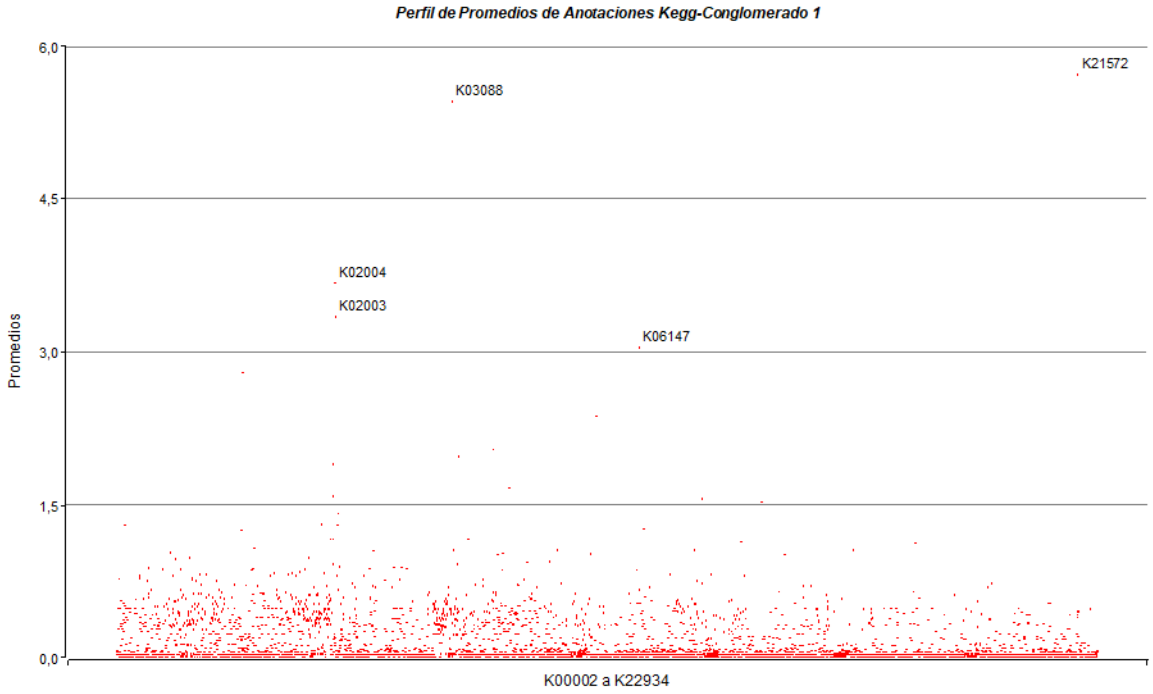
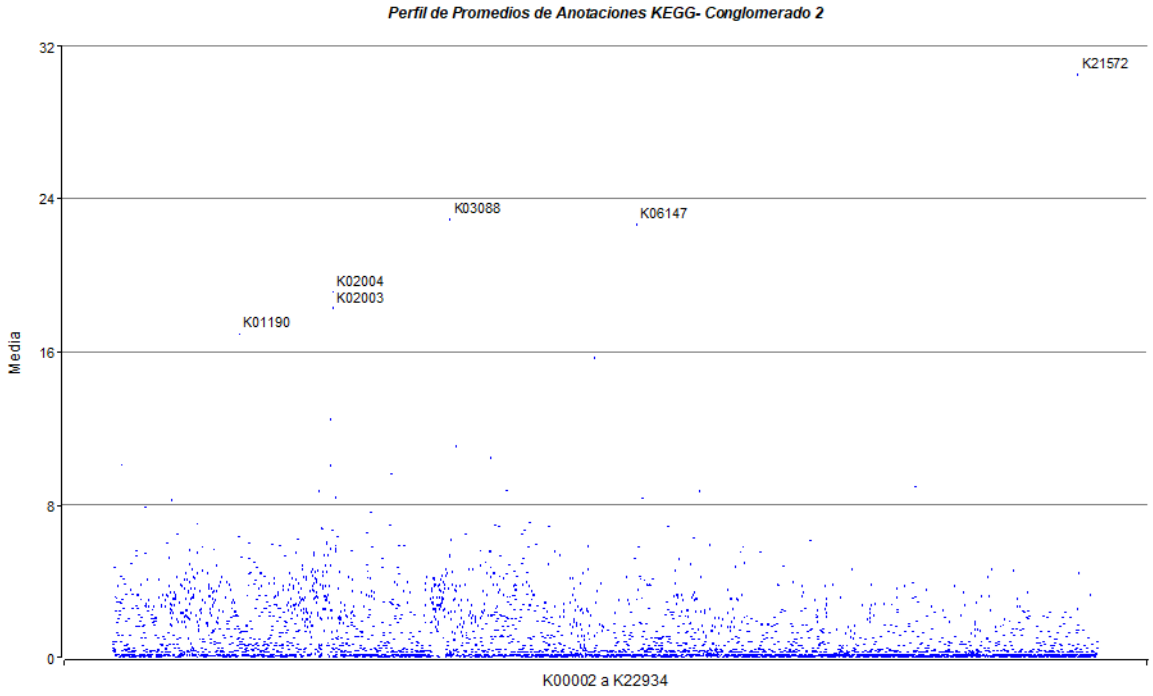


Figura 16. Perfil de Promedios de Anotaciones Kegg-Conglomerado 1



- Se estudió la influencia de cada variable en la asignación del conglomerado respectivo a través del algoritmo de selección de atributos ya citado. Los resultados pueden verse en la Tabla 6.
- El conjunto de microbiomas se particionó para obtener un conjunto de entrenamiento y otro de testeo a fin de establecer un árbol de decisión que permita clasificar cualquier microbioma, aún no estudiado, dentro uno de los conglomerados establecidos. Las Tablas 9 y 10 muestran el desempeño del ensamble Random Forest en ambos casos.

Tabla 9. Entrenamiento Ensamble Random Forest Anotaciones KO

```
=== Run information ===

Scheme:       weka.classifiers.trees.RandomForest
Relation:     Entrenamiento.KO-
Instances:    39
Attributes:   4000

Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner


=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          38               97.4359 %
Incorrectly Classified Instances         1                2.5641 %


=== Detailed Accuracy By Class ===

TP Rate  FP Rate  ROC Area  Class
1,000    0,167    1,000     a
0,833    0,000    1,000     b


=== Confusion Matrix ===

  a  b   <-- classified as
 33  0 |  a = a
  1  5 |  b = b
```

Tabla 10. Testeo Random Forest Anotaciones KO

```
=== Run information ===
```

```
Scheme:       weka.classifiers.trees.RandomForest
Instances:    37
Attributes:   4000

Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

RandomForest

Bagging with 100 iterations and base learner

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          35               94.5946 %
Incorrectly Classified Instances         2                5.4054 %
=== Detailed Accuracy By Class ===

TP Rate  FP Rate   ROC Area  Class
 1,000    0,333     0,995     a
 0,667    0,000     0,995     b


=== Confusion Matrix ===

  a  b   <-- classified as
 31  0 |  a = a
  2  4 |  b = b
```

9. Datos de pacientes autóctonos

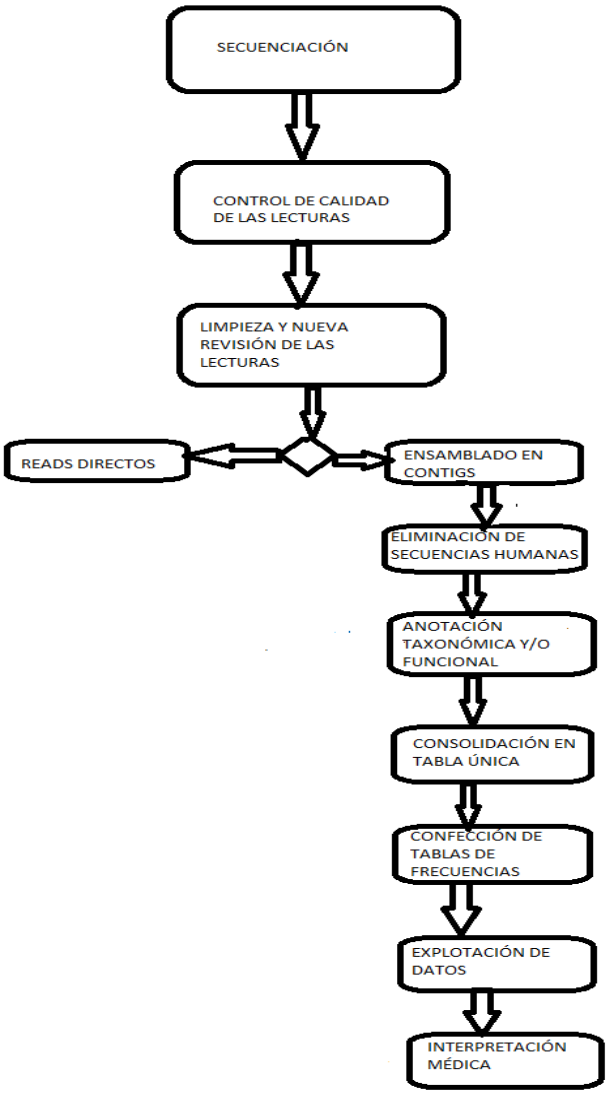
Las distintas vicisitudes presupuestarias de las que ya se dio cuenta en el informe de avance demoraron y terminaron suspendiendo la posibilidad de recoger datos de pacientes locales en el Instituto de Investigaciones Médicas Alfredo Lanari de la Universidad de Buenos Aires. Sin embargo, durante los últimos meses se pudo entrar en contacto con el Sector de Coloproctología del Hospital Italiano de Buenos Aires. Fruto de esta tarea resultó un convenio ya aprobado por el Hospital Italiano y a la firma del Rector de la UNLAM. En este marco se lograron colectar 20 muestras de materia fecal de pacientes del hospital, se adquirieron los reactivos para la secuenciación y se han enviado a secuenciar en la UNNOBA, institución que también se integrará al estudio. Así se recolectaron muestras de 10 pacientes con cáncer colorectal y otras 10 de personas sanas que formarán parte del material que se utilizará en la continuación de esta línea de investigación con el próximo proyecto.

RESULTADOS

El primer resultado importante del trabajo es el establecimiento de una pipeline de procesos a realizar, con el conocimiento acabado de las técnicas y programas involucrados, con la confección de distintos programas que unen las diferentes partes y con la experiencia

acerca de los resultados a obtener cuando se la aplica. Este es un punto significativo pues por lo general las distintas metodologías empleadas o no se revelan claramente en los artículos o se lo hace fragmentariamente de modo tal que resultaba imperioso a efecto del proceso de muestras propias. A su vez este conocimiento obtenido permitirá introducir mejoras en los procesos estudiando con mayor detalle sus variantes posibles tanto desde el punto de vista algorítmico como del de programación. El Diagrama 1 da cuenta a modo de resumen del flujo que deben seguir los datos desde que salen desde el secuenciador hasta que están en condiciones de prestarse a interpretación médica.

Diagrama 1. Flujo de Datos



Corresponde hacer algunos comentarios sobre la explotación de los datos realizada. A nivel taxonómico Phylum, tal como lo muestra la Figura 1, se detectaron 4 categorías de gran abundancia relativa en los pacientes: Actinobacteria, Bacteroidetes, Firmicutes y Proteobacteria. A su vez con abundancia relativa media se encontraron Cianobacteria y Euriarcheota. El resto presentó una baja abundancia relativa, incluso el Phylum Fusobacteria cuya especie Fusobacteria Nucleatum es prevalente en el cáncer colorectal. Los dos conglomerados obtenidos a partir del conjunto de microbiomas (Figura 2) no revelan diferencias esenciales en los promedios de frecuencia por Phylum respecto del perfil total como tampoco lo hicieron para las otras medidas de resumen que fueron consideradas. Ver Figuras 3 y 4. Se realizó el análisis de componentes principales con el objetivo de reducir las variables y se encontró que la representación gráfica de los conglomerados sobre el plano de ambas componentes principales (Figura 5) revela que los microbiomas del primer conglomerado están más concentrados y corresponden a valores más pequeños del eje principal CP1. En cambio los del segundo conglomerado están mucho más dispersos y corresponden por lo general a valores más altos de CP1 y en varios casos menores valores de CP2. A la luz de los perfiles de correlación obtenidos con estas componentes principales (Figuras 7 y 8) que explican en un 75% los valores Phylum y teniendo en cuenta que la segunda componente principal exhibió una correlación lineal negativa, es decir inversa, respecto de algunos ejes Phylum de alto perfil promedio como las Fusobacterias, en forma indirecta se estaría mostrando que los microbiomas del segundo conglomerado resultan más abundantes en términos ecológicos a nivel Phylum. En la búsqueda de hallar causas a las diferencias en la asignación de conglomerados se estudió la correlación entre las variables Phylum y la Conglomerado. La más baja correlación con ésta correspondió a Canditaus Diapherotrites (Figura 6). Además, aplicando métodos de selección de atributos respecto de la variable de clasificación Conglomerado se obtuvieron en la Tabla 4 los 10 Phylum más importantes para la clasificación. Las Figuras 9,10, 11 y 12 revelan la correlación entre cada una de las variables Phylum y las componentes CP1 y CP2 por conglomerado con la idea de avanzar en la interpretación médica de los resultados. Esto se hizo muy dificultoso ante los problemas de orden presupuestario y organizativo externos al grupo de trabajo que ya fueron citados. Por ello se prefirió no aventurar mayores conclusiones en esta etapa del trabajo propuesto y dedicar los esfuerzos a la descripción estadística y a poner a punto metodologías de aprendizaje automático que interesaran a la profesión médica en el uso de las técnicas de la metagenómica. Así se continuó con el modelado por medio de árboles de decisión de la clasificación microbiómica a nivel Phylum. Las Tablas 5 y 6 muestran el desempeño de un modelo de árbol de decisión J48 en su faz de entrenamiento y durante el testeo donde el porcentaje de microbiomas clasificados correctamente en su conglomerado fue de 90.7%. Es decir; sin que se tuviera en cuenta el valor de la variable conglomerado el algoritmo la calculó y acertó ese valor en el 90.7 % de los casos. Con ser un buen resultado fue mejor el obtenido por el modelo de ensamble de árboles denominado Random Forest que al ser testeado tuvo un porcentaje del 100% de casos bien clasificados. (Tablas 7 y 8)

También se analizaron los microbiomas según las funcionalidades KO. La Figura 13 muestra el perfil de promedios de frecuencia absoluta registrado entre sobre todo el conjunto. Se destacan algunas anotaciones KO de alta frecuencia absoluta. A título de ejemplo se cita aquí KO21572 como la de máxima frecuencia media observada.

Consultada la Kyoto Encyclopedia of Genes and Genomes (KEGG) se reproduce a continuación en la Tabla 11 la entrada a título ilustrativo.

Tabla 11. Detalle de la Anotación KO21572

ORTHOLOGY: K21572

Help

Entry	K21572 KO
Name	susD
Definition	starch-binding outer membrane protein, SusD/RagB family
Brite	KEGG Orthology (KO) [BR:ko00001] 09180 Brite Hierarchies 09183 Protein families: signaling and cellular processes 02000 Transporters K21572 susD; starch-binding outer membrane protein, SusD/RagB family Transporters [BR:ko02000] Other Transporters Accessory factors involved in transport K21572 susD; starch-binding outer membrane protein, SusD/RagB family BRITE hierarchy
Other DBs	TC: 8.A.46.1 8.A.46.3
Genes	PBW: D172_015540 PART: PARC_a0520 AAL: EP13_14395 AAUS: EP12_14960 ASP: AOR13_477 ASQ: AVL57_16105 AAW: AVL56_15025 ALE: AV939_15045 SALM: D0Y50_08170

All links

Ontology (4)

KEGG BRITE (2)

TC (2)

Gene (343686)

KEGG GENES (6316)

KEGG MGENES (311680)

RefGene (25666)

OC (24)

Protein seque

	CELL: CBR65_04170 » show all Taxonomy KOALA UniProt
Reference	PMID:10986238
Authors	Shipman JA, Berleman JE, Salyers AA
Title	Characterization of four outer membrane proteins involved in binding starch to the cell surface of Bacteroides thetaiotaomicron.
Journal	J Bacteriol 182:5365-72 (2000) DOI:10.1128/JB.182.19.5365-5372.2000
Reference	PMID:9006015
Authors	Reeves AR, Wang GR, Salyers AA
Title	Characterization of four outer membrane proteins that play a role in utilization of starch by Bacteroides thetaiotaomicron.
Journal	J Bacteriol 179:643-9 (1997) DOI:10.1128/JB.179.3.643-649.1997
Sequence	[bth:BT_3701]

nce (5098)

UniProt (5097)

SWISS-PROT (1)

Literature (2)

PubMed (2)

All databases (348790)

Download RDF

DBGET integrated database retrieval system

La información de la Tabla 11 es la que permite comprender la vía metabólica que a nivel celular puede estar presente y en términos médicos asociar ese proceso con la condición clínica del paciente. También en el caso de las Anotaciones KO se obtuvieron dos conglomerados por el método jerárquico (Figura 14) y se graficaron sus perfiles de promedios de frecuencias absolutas que resultaron similares. (Figuras 15 y 16). Finalmente se entrenó un ensamble random forest cuyo desempeño resultó muy bueno. Para el conjunto de testeo el modelo arrojó un porcentaje de casos bien clasificados del 94.6% como puede verse en la Tabla 10. Como ocurrió con los árboles para las variables Phylum hay que hacer notar la escasa diferencia obtenida entre el área bajo la curva ROC en entrenamiento que resultó 1.000 (Tabla 9) y el mismo dato en condiciones de testeo que fue de 0.995. Esto implica que el ensanble es sumamente

apto para la predicción y que su armado no se sobreajusta al conjunto de entrenamiento usado.

Tal cual se recoge en la vasta y actual bibliografía sobre el tema la metagenómica puede colaborar en el estudio del cáncer colorectal ayudando a desentrañar la etiología de algunos de sus procesos al buscar la caracterización adecuada del microbioma, su riqueza y diversidad. También es posible que en algún momento alcance a transformarse en una herramienta auxiliar al diagnóstico y a la evaluación del estadío de la enfermedad. Sin embargo, toda esta potencialidad depende en gran medida de que sea ajustada la interrelación entre lo bioinformático y lo médico. Cada algoritmo a utilizar, cada parámetro a ajustar, requieren de una evaluación acerca del grado en que colaboran a mejorar en términos médicos la herramienta de análisis. En este estudio se ha tenido cuidado en no aplicar programas como una caja negra donde entran datos y sale información. Se lo ha hecho así en la certeza de que la interpretación correcta de una información requiere algún conocimiento conceptual acerca de cómo se ha obtenido. Por supuesto se trata de un campo interdisciplinario en el cual las preguntas y las respuestas pueden venir de distintos especialistas. En este caso se ha visto que en la formación computacional de los conglomerados intervienen muy distintos aspectos que pueden modificar su composición significativamente. El primero es si se debe utilizar un método jerárquico o uno como k-means. A nivel Phylum se observó que resultaba la misma clasificación. pero no se realizó aún tal comprobación a nivel de Anotaciones KO. En segundo término, para formar los dos clusters se utilizó el encadenamiento promedio. Una tarea a efectuar es probar con los distintos tipos de encadenamiento posible. La tercera cuestión es la forma de medir la distancia. Se usó aquí la común distancia euclídea pero habría que probar la efectividad de la distancia entre distribuciones de probabilidad dado que son frecuencias absolutas o relativas los valores que las distintas variables adoptan [18]. En cada caso es el mayor o menor poder de revelar la información de tipo médico, el que debe guiar la elección que entonces ya no depende del criterio estadístico o informático solamente. Otro asunto importante lo constituye la preparación de un paquete unificado de programas, escritos en un mismo tipo de código. Aquí varios de los programas necesarios fueron confeccionados en lenguaje R, otros en cambio se escribieron en C, también se ensayó algo con el lenguaje Python y, por supuesto se aplicó una serie de programas ya realizados de código abierto o en versiones liberadas. Cada especialista conoce software relativo a su ejercicio profesional pero automatizar toda la tarea involucrada en la pipeline desarrollada sería más fácil si se la unifica en alguna forma encadenando computacionalmente los procesos que pueden ir invocándose en serie.

El detalle aportado hasta aquí da cuenta del cumplimiento de las actividades propuestas al inicio del proyecto cuyo Diagrama de Gantt se reproduce en el Diagrama 2

Diagrama 2. Diagrama de Gantt

Actividades a cumplimentar	1er Semestre	2do Semestre	3er Semestre	4to Semestre
Recolección de Datos desde repositorios internacionales.	X	X		

Santa Maria, C- López- Soria- Santa María, V-Gilardenghi				
Recolección de Muestras de Pacientes Locales y Secuenciación Santa María, C, Soria, Santa María, V		X	X	
Diseño de Experimentos Computacionales Santa María, C, Lopez – Otaegui- Avila	X	X	X	
Análisis de datos de Cáncer de Colon y Enfermedad de Crohn Santa María, C, Soria, Santa María, V	X	X	X	
Programación y Ejecución de Algoritmos Lopez-Otaegui-Martinez-Cacho Mendoza-Gilardenghi	X	X	X	X
Evaluación estadística de resultados Santa María, C, Avila, Santa María, V		X	X	X
Evaluación Biológica y Clínica de Resultados Santa María, C, Soria, Santa María, V		X	X	X
Conclusiones y Redacción Informe Final, Protocolos y Guías de Uso Santa Maria, C- López- Soria- Santa María, V-Gilardenghi				X

Cada una de las etapas señaladas ha podido desarrollarse aunque se han presentado dificultades y también puntos de avance del conocimiento donde resultó necesario decidir la continuidad de una vía de trabajo dejando de lado otra como se ha comentado. Se exponen a continuación los resultados referidos a publicaciones, formación de recursos humanos y transferencia de conocimiento.

a) Difusión y Publicaciones

Se adjuntan en Anexo IV las publicaciones y presentaciones realizadas cuyo detalle es:

- Santa María, C y Soria M. Evaluation of Richness in Microbial Communities. REDDI. Vol 2. N° 1 .2017
- Santa María, C; Santa María, V; Ávila, L; López, L; Otaegui J y Soria, M. Minería de Datos para Análisis del Microbioma Humano. WICC 2017.
- Santa María, C. Aplicaciones de Data Mining al Estudio del Microbioma. V JCC&BD. 2017
- Santa María, C. Grafos para Reconocimiento de Patrones. I Workshop de Reconocimiento de Patrones. UNLaM .2017
- Santa María, C ; Rebrij, Romina, Santa María, Victoria; Lopez, Luis y Soria, Marcelo. Reconocimiento de Patrones Genéticos por Medio de Grafos. WICC 2018
- Santa María, C; Rebrij, R; Santa María, V y Soria, M. Treatment of Massive Metagenomic Data with Graphs. VIJCC&BD. 2018

b) Formación de Recursos Humanos

Bajo la dirección del Dr. Marcelo Soria ya desarrolla su tesis de doctorado en ingeniería en la Facultad de Ingeniería de UBA la Ing. Romina Rebrij quien trabaja en el sector de Informática del Hospital Italiano y ha participado en el trabajo y las publicaciones llevadas adelante. También la Lic. Laura Ávila está desarrollando su tesis de maestría en Informática en UNLAM bajo la dirección del Mg. Cristóbal Santa María.

c) Transferencia

Un aspecto que ha limitado el trabajo ha sido la dificultad que por distintos motivos se ha tenido en contar con especialistas médicos en gastroenterología y cáncer que cumplieran con la etapa final de evaluación médica de los resultados. En tal sentido abre una perspectiva promisoria el convenio que está a la firma entre la UNLAM y el Hospital Italiano de Buenos Aires para trabajar en el tema. A partir de esta reciente vinculación médica, más ampliamente especializada, se ha focalizado el trabajo sobre pacientes con cáncer de colon y se ha preferido dejar para más adelante el análisis microbiómico respecto a la enfermedad de Crohn. Se espera además una mayor interrelación con los investigadores médicos y una orientación general del trabajo del grupo más ajustada a sus criterios y necesidades clínicas.

VINCULACIÓN CON OTROS GRUPOS DE INVESTIGACIÓN

Como se ha comentado más arriba se ha logrado establecer un vínculo formal entre el grupo de investigación y el Sector de Coloproctología del Hospital Italiano de Buenos Aires con el que se espera continuar en la misma línea de trabajo en proyectos futuros. Existe también una vinculación con otros grupos e instituciones a través de la participación en el grupo de UNLAM, en carácter de Investigador Externo, del Dr. Marcelo Soria, Director de la Maestría en Explotación de Datos y Descubrimiento del Conocimiento de la FCEyN y Profesor de la Cátedra de Microbiología en la FA ambas de UBA. Además, participó en la investigación, en carácter de asesora externa, la médica Victoria Santa María, especialista en estadística médica y cirujana de planta del Instituto Lanari de la Facultad de Medicina de la UBA lo que permitió el nexo con esa institución.

CONCLUSIONES

Se ha podido establecer una pipeline con varios pasos automatizados para tratar las secuencias de ADN microbiómico desde que salen del secuenciador hasta que resultan datos para explotación por técnicas estadísticas multivariadas y de aprendizaje supervisado y no supervisado, de tal forma de ponerlos al servicio de la interpretación médica. Se ha logrado también tomar muestras de pacientes autóctonos y firmar un convenio de colaboración que abre las puertas para el uso médico en el medio local de la tecnología hasta aquí desarrollada y/o utilizada. En la continuidad de la línea de investigación se espera lograr un marco más amplio de validación local e internacional para el trabajo.

REFERENCIAS

[1] Lopez, A et al. Microbiota in digestive cancers: our new partner? Carcinogenesis, 2017, 1-10. doi:10.1093/carcin/bgx087

[2] Castellarin, M et al. Fusobacterium nucleatum infection is prevalent in human colorectal carcinoma.Genome Research. Pp. 299-306. 2012

[3] https://www.ncbi.nlm.nih.gov/bioproject/?term=PRJNA397450

[4] Jones, R B. et al. Inter-niche and inter-individual variation in gut microbial community assessment using stool, rectal swab and mucosal samples. Scientific Reports volume 8, Article number: 4139 (2018) www.nature.com/scientificreports

[5] https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

[6] https://github.com/vsbuffalo/scythe

[7] https://cutadapt.readthedocs.io/en/stable/

[8]Coil, D; Jospin, G; and Darling, A. A5-miseq: an updated pipeline to assemble microbial genomes from Illumina MiSeq data. Bioinformatics. Vol. 31 n° 4 587-589. (2015)

[9] Peng, Y; Leung, H C M; Yiu, S M; Chin F Y L. IDBA-UD: de novo assembler for single-cell and metagenomic sequencing data with hibhly uneven depth. Bioinformatics. Vol. 28. n° 11. 1420-1428. (2012)

[10] https://academic.oup.com/bioinformatics/article/31/10/1674/177884)

Li, D; Liu, CM; Luo, R; Sadakane, K and Lam, TW. MEGAHIT: an ultra-fast single-node solution for large and complex metagenomics assembly via succinct de Bruijn graph. Bioinformatics. Vol. 31 n°10. 1674-1676 (2015)

[11] https://github.com/voutcn/megahit

[12] http://kaiju.binf.ku.dk/

[13] https://www.nature.com/articles/ncomms11257.

[14] https://www.genome.jp/kegg/kaas/

[15] http://bowtie-bio.sourceforge.net/bowtie2/ y

https://www.nature.com/articles/nmeth.1923

[16] M. Kanehisa, Y. Sato; M. Furumichi , K. Morishima y M. Tanabe. New approach for understanding genome variations in KEGG. Nucleic Acids Research, Volume 47, Issue D1, 8 January 2019, Pages D590–D595, https://doi.org/10.1093/nar/gky962 [17] https://www.cs.waikato.ac.nz/ml/weka/

[18] http://www.infostat.com.ar/

[19] M. A. Hall. Correlation-based Feature Subset Selection for Machine Learning. Hamilton. New Zeland. 1988

[20] Statnikov A. Henaff M. Narendra V. Konganti K. Li Z. Yang L. Pei Z. Blaser M. AliferisC y Alekseyenko A. (2013) A comprehensive evaluation of multicategory classification methods for microbiomic data. Microbiome 2013 1:11

[21] Breiman, Leo (2001). «Random Forests». Machine Learning **45** (1): 5–32. doi:10.1023/A:1010933404324.

[22] Endres, D y Schindeling,J. A New Metric for Probability Distributions. IEEE.

Transactions on Information Theory. Vol.49 NO.7. 2003.

APÉNDICE

Programa C Microbioma.Phylum

main.h

```
#ifndef MAIN_H_
#define MAIN_H_

#include <stdio.h>
#include <stdlib.h>

#include "funciones.h"


#define     NOM_SAL         "informe.csv"
#define     NOM_SAL         "informe.csv"


#endif
```

main.c

```
#include "main.h"

int main(void)
{
    FILE   *fpArchivos,
           *fpExtra,
           *fpSalida;
    tLista  listaK;
    int     generados = 0;

    crearLista(&listaK);
///
    system("del " PATH_CSV "\\*.csv");
    system("dir " PATH_CSV_EXTRA "\\*.csv /b /on > extra.txt");
    if(!abrirArchivo(&fpExtra, "extra.txt", "rt", CON_MSJ))
        return 1;
    generados = generarDatos(fpExtra);
    fclose(fpExtra);
    if(generados == 0)
    {
        fprintf(stderr, "ERROR - no se pudieron convertir los archivos\n");
        return 1;
    }
///


    system("dir " PATH_CSV "\\*.csv /b /on > archivos.txt");
    if(!abrirArchivo(&fpArchivos, "archivos.txt", "rt", CON_MSJ))
        return 1;
    if(!abrirArchivo(&fpSalida, NOM_SAL, "wt", CON_MSJ))
    {
        fclose(fpSalida);
        return 2;
    }
    leerTxt(fpArchivos, &listaK, 1, fpSalida);

    rewind(fpArchivos);
```

```
    leerTxt(fpArchivos, &listaK, 0, fpSalida);

    vaciarLista(&listaK);
    fclose(fpArchivos);
    fclose(fpSalida);
    return 0;
}
```

funciones.h

```
#ifndef FUNCIONES_H_
#define FUNCIONES_H_

#include <stdio.h>
#include <string.h>

#include "lista.h"


typedef struct
{
/// char claveK[20];
    char clavePh[40];
    int  claveFA;
} tInfo;

int compararXClaPh(const void *d1, const void *d2);

void acumClaFA(void *d1, const void *d2);

#define     CON_MSJ         1
int abrirArchivo(FILE **fo, const char *nom, const char *modo, int conSin);

#define PATH_CSV_EXTRA      "..\\datos\\extra_datos"
int generarDatos(FILE *fp);
int leerExtraGrabarCSV(FILE *fpSal, FILE *fpEnt);

#define PATH_CSV            "..\\datos"
int leerTxt(FILE *fpEnt, tLista *p, int priVez, FILE *fpSalida);

int leerCSV(const char *nomCSV, tLista *p, int priVez);

int generarPrimeraLinea(tLista *p, FILE *fpSalida);

int generarDetalle(tLista *pK, tLista *pFA, FILE *fpSalida);


#endif
```

funciones.c

```
#include "funciones.h"

int abrirArchivo(FILE **fp, const char *nom, const char *modo, int conSin)
{
    *fp = fopen(nom, modo);
    if(*fp == NULL)
```

```
    {
        if(conSin == CON_MSJ)
            fprintf(stderr,
                    "ERROR - abriendo \"%s\" en modo \"%s\"\n",
                    nom, modo);
        return 0;
    }
    return 1;
}

/** PRIMERA VEZ
 ** lee el archivo (generado en main), con los nombres de archivos a procesar
 ** lee cada uno de los archivos (claveK) para almacenarlos en una lista
 **     ordenada (llevando cuenta de las veces que aparece cada claveK)
 **     para generar la primera línea (y segunda) del informe
 ** SEGUNDA VEZ
 ** lee el archivo (generado en main), con los nombres de archivos a procesar
 ** lee cada uno de los archivos (claveK y claveFA) para almacenarlos en otra
 **     lista ordenada para luego, por cada archivo generar la línea del
informe
 **     con nombre de archivo y valor de la claveFA (completando con 0 -cero-
 **     las posiciones que no tienen una claveK-claveFA)
 **     para generar la primera línea (y segunda) del informe
 **/
int leerTxt(FILE *fpEnt, tLista *p, int priVez, FILE *fpSalida)
{
    char    linea[300],
           *aux;
    int     cont = 0;
    tLista  listaFA;

    if(!priVez)
        crearLista(&listaFA);

    while(fgets(linea, sizeof(linea), fpEnt))
    {
        if((aux = strchr(linea, '\n')) == NULL)
            return fprintf(stderr, "ERROR - 001\n") & 0;
        *aux = '\0';
        if(priVez)
            leerCSV(linea, p, priVez);
        else
        {
            char    nombreMuestra[300],
                   *aux;
            strcpy(nombreMuestra, linea);
            if((aux = strchr(nombreMuestra, '.')) != NULL)
                *aux = '\0';
            fprintf(fpSalida, "%s", nombreMuestra);
            leerCSV(linea, &listaFA, priVez);
            generarDetalle(p, &listaFA, fpSalida);
            /// ***
        }
        cont++;
    }
    printf("Se %s %d archivos\n", priVez ? "leyeron" : "procesaron", cont);
```

```
    if(priVez)
        generarPrimeraLinea(p, fpSalida);
    return cont;
}


/** en la primera vez que se la invoca carga una lista con las claveK
 *      de todos los archivos y acumula las veces que aparece esa claveK
 *  las siguientes veces se invoca con otra lista y guarda claveK y claveFA
 *  devuelve la cantidad de registros (sin contar el primero)
 **/
int leerCSV(const char *nomCSV, tLista *p, int priVez)
{
    FILE   *fpCSV;
    char    linea[300],
           *aux,
            patNamCSV[300];
    tInfo   info;
    int     cantReg = 0;

    sprintf(patNamCSV, "%s\\%s", PATH_CSV, nomCSV);
    if(!abrirArchivo(&fpCSV, patNamCSV, "rt", CON_MSJ))
        return 0;
    if(!fgets(linea, sizeof(linea), fpCSV) ||
       (aux = strchr(linea, '\n')) == NULL)
        return fprintf(stderr, "ERROR - 002\n") & 0;
    while(fgets(linea, sizeof(linea), fpCSV))
    {
        if((aux = strchr(linea, '\n')) == NULL)
            return fprintf(stderr, "ERROR - 002Bis\n") & 0;
        *aux = '\0';
        if((aux = strchr(linea, '\t')) == NULL)
            return fprintf(stderr, "ERROR - 002Bis\n") & 0;
        sscanf(aux + 1, "%d", &info.claveFA);
        *aux = '\0';
        *info.clavePh = '\0';
        if(!strcmpi(linea, "NA"))
            strcat(info.clavePh, "|");
        strcat(info.clavePh, linea);
        if(priVez == 1)
            info.claveFA = 1;
        if(insertarEnOrden(p, &info, sizeof(info),
                           compararXClaPh, acumClaFA) == SIN_MEM)
            return fprintf(stderr, "ERROR - sin memoria\n") & 0;
        cantReg++;
    }

    fclose(fpCSV);
    return cantReg;
}

int compararXClaPh(const void *d1, const void *d2)
{
    return strcmpi(((tInfo*)d1)->clavePh, ((tInfo*)d2)->clavePh);
}
```

```c
void acumClaFA(void *d1, const void *d2)
{
    ((tInfo*)d1)->claveFA += ((tInfo*)d2)->claveFA;
}

int generarPrimeraLinea(tLista *p, FILE *fpSalida)
{
    if(irAlPrimero(p))
    {
        tInfo info;
        do
        {

            if(verActual(p, &info, sizeof(tInfo)))
                fprintf(fpSalida,
                        "\t%s",
                        info.clavePh + (*info.clavePh == '|'));
        } while(irAlSiguiente(p));
        fprintf(fpSalida, "\n");
        irAlPrimero(p);
        do
        {
            if(verActual(p, &info, sizeof(tInfo)))
                fprintf(fpSalida, "\t%d", info.claveFA);
        } while(irAlSiguiente(p));
        fprintf(fpSalida, "\n");
        return 1;
    }
    return 0;
}

int generarDetalle(tLista *pK, tLista *pFA, FILE *fpSalida)
{
    tInfo   infoPh,
            infoFA;

    if(!irAlPrimero(pK) || !irAlPrimero(pFA))
        return 0;
    while(verActual(pK, &infoPh, sizeof(infoPh)))
    {
        infoFA = infoPh;
        if(!buscarYEliminar(pFA, &infoFA, sizeof(infoFA), compararXClaPh))
            infoFA.claveFA = 0;
        fprintf(fpSalida, "\t%d", infoFA.claveFA);
        if(!irAlSiguiente(pK))
            break;
    }
    if(!listaVacia(pFA))
        return fprintf(stderr, "ERROR - catastrófico 003\n") & 0;
    fprintf(fpSalida, "\n");

    return 1;
}

/** leyendo cada archivo del subdirectorio datos\datos_extra genera
 **     cada archivo en el subdirectorio \datos con las dos columnas de interés
```

```
 **      Categorías y FA separados por \t
 **/
int generarDatos(FILE *fp)
{
    int          cant = 0;
    char         linea[300],
                 nomEnt[300],
                 nomSal[300],
                *aux;
    FILE        *fpEnt,
                *fpSal;

    while(fgets(linea, sizeof(linea), fp))
    {
        if((aux = strchr(linea, '\n')) != NULL)
            *aux = '\0';
        sprintf(nomEnt, "%s\\%s", PATH_CSV_EXTRA, linea);
        sprintf(nomSal, "%s\\%s", PATH_CSV, linea);
        if(!abrirArchivo(&fpEnt, nomEnt, "rt", CON_MSJ))
            return 0;
        if(!abrirArchivo(&fpSal, nomSal, "wt", CON_MSJ))
        {
            fclose(fpEnt);
            return 0;
        }
        if(!leerExtraGrabarCSV(fpSal, fpEnt))
           return 0;
        fclose(fpEnt);
        fclose(fpSal);
        cant++;
    }
    printf("Se convirtieron %d archivos\n", cant);
    return cant;
}

int leerExtraGrabarCSV(FILE *fpSal, FILE *fpEnt)
{
    char         linea[300],
                *aux;

    while(fgets(linea, sizeof(linea), fpEnt))
    {
        if((aux = strrchr(linea, ';')) == NULL)
            return 0;
        *aux = '\0';
        if((aux = strrchr(linea, ';')) == NULL)
            return 0;
        *aux = '\t';
        if((aux = strrchr(linea, ';')) == NULL)
            return 0;
        fprintf(fpSal, "%s\n", aux + 1);
    }
    return 1;
}
```

lista.h

```c
#ifndef LISTA_H_
#define LISTA_H_

#include <string.h>
#include <stdlib.h>


#define     TODO_BIEN       1
#define     CLA_DUP         2
#define     SIN_MEM         3

typedef struct sNodo
{
    void            *info;
    unsigned         tamInfo;
    struct sNodo    *sig;
    struct sNodo    *ant;
} tNodo, *tLista;

#define   menor( X , Y )    ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )

void crearLista(tLista *p);
int listaVacia(const tLista *p);
int insertarEnOrden(tLista *p, const void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *),
                    void (*acum)(void *, const void *));
int irAlPrimero(tLista *p);
int irAlSiguiente(tLista *p);
int verActual(tLista *p, void *info, unsigned tamInfo);
int vaciarLista(tLista *p);
int buscarYEliminar(tLista *p, void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *));


#endif
```

lista.c

```c
#include "lista.h"

void crearLista(tLista *p)
{
    *p = NULL;
}

int listaVacia(const tLista *p)
{
    return *p == NULL;
}

int insertarEnOrden(tLista *p, const void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *),
                    void (*acum)(void *, const void *))
{
    tNodo       *ant,
                *sig,
```

```
                *act = *p,
                *nue;
    int          cmp;

    if(act == NULL)
    {
        ant = NULL;
        sig = NULL;
    }
    else
    {
        while(act->sig && comp(act->info, info) < 0)
            act = act->sig;
        while(act->ant && comp(act->info, info) > 0)
            act = act->ant;
        cmp = comp(act->info, info);
        if(cmp == 0)
        {
            if(acum)
                acum(act->info, info);
            *p = act;
            return CLA_DUP;
        }
        if(cmp > 0)
        {
            ant = act->ant;
            sig = act;
        }
        else
        {
            ant = act;
            sig = act->sig;
        }
    }
    nue = (tNodo *)malloc(sizeof(tNodo));
    if(nue == NULL)
        return SIN_MEM;
    nue->info = malloc(tamInfo);
    if(nue->info == NULL)
    {
        free(nue);
        return SIN_MEM;
    }
    memcpy(nue->info, info, tamInfo);
    nue->tamInfo = tamInfo;
    nue->sig = sig;
    nue->ant = ant;
    if(ant)
        ant->sig = nue;
    if(sig)
        sig->ant = nue;
    *p = nue;
    return TODO_BIEN;
}

int irAlPrimero(tLista *p)
```

```
{
    tNodo *act = *p;
    if(act)
    {
        while(act->ant)
            act = act->ant;
        *p = act;
        return 1;
    }
    return 0;
}

int irAlSiguiente(tLista *p)
{
    if((*p)->sig)
    {
        *p = (*p)->sig;
        return 1;
    }
    return 0;
}

int verActual(tLista *p, void *info, unsigned tamInfo)
{
    if(*p)
    {
        memcpy(info, (*p)->info, menor(tamInfo, (*p)->tamInfo));
        return 1;
    }
    return 0;
}

int vaciarLista(tLista *p)
{
    tNodo  *act = *p;
    int     cant = 0;
    if(act)
    {
        while(act->sig)
        {
            tNodo *aux = act->sig;
            act->sig = aux->sig;
            free(aux->info);
            free(aux);
            cant++;
        }
        while(act->ant)
        {
            tNodo *aux = act->ant;
            act->ant = aux->ant;
            free(aux->info);
            free(aux);
            cant++;
        }
        free(act->info);
        free(act);
```

```
        cant++;
        *p = NULL;
    }
    return cant;
}

int buscarYEliminar(tLista *p, void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *))
{
    tNodo *act = *p;

    if(act == NULL)
        return 0;
    while(act->ant)
        act = act->ant;
    while(act && comp(act->info, info) < 0)
        act = act->sig;
    if(act && !comp(act->info, info))
    {
        tNodo  *ant = act->ant,
               *sig = act->sig;

        memcpy(info, act->info, menor(tamInfo, act->tamInfo));
        free(act->info);
        free(act);
        if(sig)
        {
            *p = sig;
            sig->ant = ant;
        }
        else
            *p = ant;
        if(ant)
            ant->sig = sig;
        return 1;
    }
    return 0;
}
```

Programa C Microbiomas.KO

main.h

```
#ifndef MAIN_H_
#define MAIN_H_

#include <stdio.h>
#include <stdlib.h>
```

```
#include "funciones.h"


#define     NOM_SAL         "informe.csv"
#define     NOM_SAL         "informe.csv"


#endif
```

main.c

```
#include "main.h"

int main(void)
{
    FILE   *fpArchivos,
           *fpSalida;
    tLista  listaK;

    crearLista(&listaK);

    system("dir " PATH_CSV "\\*.csv /b /on > archivos.txt");
    if(!abrirArchivo(&fpArchivos, "archivos.txt", "rt", CON_MSJ))
        return 1;
    if(!abrirArchivo(&fpSalida, NOM_SAL, "wt", CON_MSJ))
    {
        fclose(fpSalida);
        return 2;
    }
    leerTxt(fpArchivos, &listaK, 1, fpSalida);

    rewind(fpArchivos);
    leerTxt(fpArchivos, &listaK, 0, fpSalida);

    vaciarLista(&listaK);
    fclose(fpArchivos);
    fclose(fpSalida);
    return 2;
}
```

funciones.h

```
#ifndef FUNCIONES_H_
#define FUNCIONES_H_

#include <stdio.h>
#include <string.h>

#include "lista.h"


typedef struct
```

```
{
    char claveK[20];
    int  claveFA;
} tInfo;

int compararXClaK(const void *d1, const void *d2);

void acumClaFA(void *d1, const void *d2);

#define     CON_MSJ         1
int abrirArchivo(FILE **fo, const char *nom, const char *modo, int conSin);

#define PATH_CSV            "..\\datos"
int leerTxt(FILE *fpEnt, tLista *p, int priVez, FILE *fpSalida);

int leerCSV(const char *nomCSV, tLista *p, int priVez);

int generarPrimeraLinea(tLista *p, FILE *fpSalida);

int generarDetalle(tLista *pK, tLista *pFA, FILE *fpSalida);


#endif
```

funciones.c

```
#include "funciones.h"

int abrirArchivo(FILE **fp, const char *nom, const char *modo, int conSin)
{
    *fp = fopen(nom, modo);
    if(*fp == NULL)
    {
        if(conSin == CON_MSJ)
            fprintf(stderr,
                    "ERROR - abriendo \"%s\" en modo \"%s\"\n",
                    nom, modo);
        return 0;
    }
    return 1;
}

/** PRIMERA VEZ
 ** lee el archivo (generado en main), con los nombres de archivos a procesar
 ** lee cada uno de los archivos (claveK) para almacenarlos en una lista
 **     ordenada (llevando cuenta de las veces que aparece cada claveK)
 **     para generar la primera línea (y segunda) del informe
 ** SEGUNDA VEZ
 ** lee el archivo (generado en main), con los nombres de archivos a procesar
 ** lee cada uno de los archivos (claveK y claveFA) para almacenarlos en otra
 **     lista ordenada para luego, por cada archivo generar la línea del
   informe
 **     con nombre de archivo y valor de la claveFA (completando con 0 -cero-
 **     las posiciones que no tienen una claveK-claveFA)
 **     para generar la primera línea (y segunda) del informe
```

```
 **/
int leerTxt(FILE *fpEnt, tLista *p, int priVez, FILE *fpSalida)
{
    char    linea[300],
           *aux;
    int     cont = 0;
    tLista  listaFA;

    if(!priVez)
        crearLista(&listaFA);

    while(fgets(linea, sizeof(linea), fpEnt))
    {
        if((aux = strchr(linea, '\n')) == NULL)
            return fprintf(stderr, "ERROR - 001\n") & 0;
        *aux = '\0';
        if(priVez)
            leerCSV(linea, p, priVez);
        else
        {
            char    nombreMuestra[300],
                   *aux;
            strcpy(nombreMuestra, linea);
            if((aux = strchr(nombreMuestra, '.')) != NULL)
                *aux = '\0';
            fprintf(fpSalida, "%s", nombreMuestra);
            leerCSV(linea, &listaFA, priVez);
            generarDetalle(p, &listaFA, fpSalida);
            /// ***
        }
        cont++;
    }
    printf("Se %s %d archivos\n", priVez ? "leyeron" : "procesaron", cont);
    if(priVez)
        generarPrimeraLinea(p, fpSalida);
    return cont;
}


/** en la primera vez que se la invoca carga una lista con las claveK
 *      de todos los archivos y acumula las veces que aparece esa claveK
 *  las siguientes veces se invoca con otra lista y guarda claveK y claveFA
 *  devuelve la cantidad de registros (sin contar el primero)
 **/
int leerCSV(const char *nomCSV, tLista *p, int priVez)
{
    FILE   *fpCSV;
    char    linea[300],
           *aux,
            patNamCSV[300];
    tInfo   infoK;
    int     cantReg = 0;

    sprintf(patNamCSV, "%s\\%s", PATH_CSV, nomCSV);
    if(!abrirArchivo(&fpCSV, patNamCSV, "rt", CON_MSJ))
        return 0;
```

```
    if(!fgets(linea, sizeof(linea), fpCSV) ||
       (aux = strchr(linea, '\n')) == NULL)
        return fprintf(stderr, "ERROR - 002\n") & 0;
    while(fgets(linea, sizeof(linea), fpCSV))
    {
        if((aux = strchr(linea, '\n')) == NULL)
            return fprintf(stderr, "ERROR - 002Bis\n") & 0;
        *aux = '\0';
        sscanf(linea, "%s\t%d", infoK.claveK, &infoK.claveFA);
///     printf("%s - %d\n", infoK.claveK, infoK.claveFA);    /// ***
        if(priVez == 1)
            infoK.claveFA = 1;
        if(insertarEnOrden(p, &infoK, sizeof(infoK),
                           compararXClaK, acumClaFA) == SIN_MEM)
            return fprintf(stderr, "ERROR - sin memoria\n") & 0;
        cantReg++;
    }

    fclose(fpCSV);
    return cantReg;
}

int compararXClaK(const void *d1, const void *d2)
{
    return strcmp(((tInfo*)d1)->claveK, ((tInfo*)d2)->claveK);
}

void acumClaFA(void *d1, const void *d2)
{
    ((tInfo*)d1)->claveFA += ((tInfo*)d2)->claveFA;
}

int generarPrimeraLinea(tLista *p, FILE *fpSalida)
{
    if(irAlPrimero(p))
    {
        tInfo info;
        do
        {

            if(verActual(p, &info, sizeof(tInfo)))
                fprintf(fpSalida, "\t%s", info.claveK);
        } while(irAlSiguiente(p));
        fprintf(fpSalida, "\n");
        irAlPrimero(p);
        do
        {
            if(verActual(p, &info, sizeof(tInfo)))
                fprintf(fpSalida, "\t%d", info.claveFA);
        } while(irAlSiguiente(p));
        fprintf(fpSalida, "\n");
        return 1;
    }
    return 0;
}
```

```c
int generarDetalle(tLista *pK, tLista *pFA, FILE *fpSalida)
{
    tInfo   infoK,
            infoFA;

    if(!irAlPrimero(pK) || !irAlPrimero(pFA))
        return 0;
    while(verActual(pK, &infoK, sizeof(infoK)))
    {
        infoFA = infoK;
        if(!buscarYEliminar(pFA, &infoFA, sizeof(infoFA), compararXClaK))
            infoFA.claveFA = 0;
        fprintf(fpSalida, "\t%d", infoFA.claveFA);
        if(!irAlSiguiente(pK))
            break;
    }
    if(!listaVacia(pFA))
        return fprintf(stderr, "ERROR - catastrófico 003\n") & 0;
    fprintf(fpSalida, "\n");

    return 1;
}
```

lista.h

```c
#ifndef LISTA_H_
#define LISTA_H_

#include <string.h>
#include <stdlib.h>


#define     TODO_BIEN       1
#define     CLA_DUP         2
#define     SIN_MEM         3

typedef struct sNodo
{
    void            *info;
    unsigned         tamInfo;
    struct sNodo    *sig;
    struct sNodo    *ant;
} tNodo, *tLista;

#define   menor( X , Y )    ( ( X ) <= ( Y ) ? ( X ) : ( Y ) )

void crearLista(tLista *p);
int listaVacia(const tLista *p);
int insertarEnOrden(tLista *p, const void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *),
                    void (*acum)(void *, const void *));
int irAlPrimero(tLista *p);
int irAlSiguiente(tLista *p);
int verActual(tLista *p, void *info, unsigned tamInfo);
```

```
int vaciarLista(tLista *p);
int buscarYEliminar(tLista *p, void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *));


#endif
```

lista.c

```
#include "lista.h"

void crearLista(tLista *p)
{
    *p = NULL;
}

int listaVacia(const tLista *p)
{
    return *p == NULL;
}

int insertarEnOrden(tLista *p, const void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *),
                    void (*acum)(void *, const void *))
{
    tNodo      *ant,
               *sig,
               *act = *p,
               *nue;
    int         cmp;

    if(act == NULL)
    {
        ant = NULL;
        sig = NULL;
    }
    else
    {
        while(act->sig && comp(act->info, info) < 0)
            act = act->sig;
        while(act->ant && comp(act->info, info) > 0)
            act = act->ant;
        cmp = comp(act->info, info);
        if(cmp == 0)
        {
            if(acum)
                acum(act->info, info);
            *p = act;
            return CLA_DUP;
        }
        if(cmp > 0)
        {
            ant = act->ant;
```

```
            sig = act;
        }
        else
        {
            ant = act;
            sig = act->sig;
        }
    }
    nue = (tNodo *)malloc(sizeof(tNodo));
    if(nue == NULL)
        return SIN_MEM;
    nue->info = malloc(tamInfo);
    if(nue->info == NULL)
    {
        free(nue);
        return SIN_MEM;
    }
    memcpy(nue->info, info, tamInfo);
    nue->tamInfo = tamInfo;
    nue->sig = sig;
    nue->ant = ant;
    if(ant)
        ant->sig = nue;
    if(sig)
        sig->ant = nue;
    *p = nue;
    return TODO_BIEN;
}

int irAlPrimero(tLista *p)
{
    tNodo *act = *p;
    if(act)
    {
        while(act->ant)
            act = act->ant;
        *p = act;
        return 1;
    }
    return 0;
}

int irAlSiguiente(tLista *p)
{
    if((*p)->sig)
    {
        *p = (*p)->sig;
        return 1;
    }
    return 0;
}

int verActual(tLista *p, void *info, unsigned tamInfo)
{
    if(*p)
    {
```

```
        memcpy(info, (*p)->info, menor(tamInfo, (*p)->tamInfo));
        return 1;
    }
    return 0;
}

int vaciarLista(tLista *p)
{
    tNodo  *act = *p;
    int     cant = 0;
    if(act)
    {
        while(act->sig)
        {
            tNodo *aux = act->sig;
            act->sig = aux->sig;
            free(aux->info);
            free(aux);
            cant++;
        }
        while(act->ant)
        {
            tNodo *aux = act->ant;
            act->ant = aux->ant;
            free(aux->info);
            free(aux);
            cant++;
        }
        free(act->info);
        free(act);
        cant++;
        *p = NULL;
    }
    return cant;
}

int buscarYEliminar(tLista *p, void *info, unsigned tamInfo,
                    int (*comp)(const void *, const void *))
{
    tNodo *act = *p;

    if(act == NULL)
        return 0;
    while(act->ant)
        act = act->ant;
    while(act && comp(act->info, info) < 0)
        act = act->sig;
    if(act && !comp(act->info, info))
    {
        tNodo  *ant = act->ant,
               *sig = act->sig;

        memcpy(info, act->info, menor(tamInfo, act->tamInfo));
        free(act->info);
        free(act);
        if(sig)
```

```
        {
            *p = sig;
            sig->ant = ant;
        }
        else
            *p = ant;
        if(ant)
            ant->sig = sig;
        return 1;
    }
    return 0;
}
```

3. Cuerpo de anexos

Anexo I Rendición de gastos (incorporado al SPI)

Anexo III Certificados (incorporado al SPI)

- 3.1 Certificado WICC2018
- 3.2 Certificado VI Jornadas de Cloud Computing
- 3.3 Certificado V Jornadas de Cloud Computing
- 3.4 Certificado Workshop Reconocimiento de Patrones
- 3.5 Certificado Asistencia WICC2017
- 3.6 Certificado Artículo WICC2017

Anexo IV Artículos y Exposiciones (incorporado al SPI)

- 4.1 Artículo Treatment of Massive Metagenomic Data with Graphs. VIJCC&BD. 2018
- 4.2 Exposición Tratamiento de Datos Metagenómicos Masivos por medio de Grafos. VI JCC&BD
- 4.3 Artículo Reconocimiento de Patrones Genéticos por Medio de Grafos. WICC 2018
- 4.4 Póster WICC2018
- 4.5 Póster WICC2017
- 4.6 Artículo Minería de Datos para Análisis del Microbioma Humano. WICC 2017.
- 4.7 Exposición Grafos para Reconocimiento de Patrones Genéticos. Workshop de Reconocimiento de Patrones.
- 4.8 Artículo Evaluation of Richness in Microbial Communities. REDDI.
- 4.9 Exposición Aplicaciones de Data Mining al Estudio del Microbioma. V JCC&BD. 2017