UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE CIENCIAS NATURALES
LICENCIATURA EN BIOLOGÍA

ANÁLISIS BIOINFORMÁTICO DE USO DE ANTICODONES Y DISTRIBUCIÓN DE FINALES DE SECUENCIAS CCA EN EL EXTREMO 3' DE LOS tRNAs BACTERIANOS.

TESIS INDIVIDUAL

Que como parte de los requisitos para obtener el grado de

Licenciado en Biología

PRESENTA:
JULIO ALFONSO CRUZ MEDINA

DIRIGIDO POR:
DR. JUAN CAMPOS GUILLÉN

QUERÉTARO, QRO.
MAYO 2009

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO
FACULTAD DE CIENCIAS NATURALES
LICENCIATURA EN BIOLOGÍA

ANÁLISIS BIOINFORMÁTICO DE USO DE ANTICODONES Y DISTRIBUCIÓN DE FINALES DE SECUENCIAS CCA EN EL EXTREMO 3' DE LOS tRNAs BACTERIANOS.

Tesis individual
Que como parte de los requisitos para obtener el grado de
Licenciado en Biología

Presenta:
Julio Alfonso Cruz Medina

Dirigido por:
Dr. Juan Campos Guillén

SINODALES

Dr. Juan Campos Guillén
Presidente

Dra. Gabriela Olmedo Álvarez
Secretario

Luis David Alcaraz Peraza
Vocal

Varinia López Ramírez
Suplente

Centro Universitario
Querétaro, Qro.
Mayo 2009
México

El presente trabajo fue realizado en los laboratorios de Microbiología de la Facultad de Ciencias Naturales a cargo del Dr. Juan Campos Guillen de la Universidad Autónoma de Querétaro y el laboratorio de Biología Molecular de Bacterias II del departamento de Ingeniería Genética, a cargo de Dra. Gabriela Olmedo Álvarez, del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional Unidad Irapuato (CINVESTAV-Irapuato).

RESUMEN

El RNA de transferencia (tRNA) es una cadena sencilla de ~76 nucleótidos que adopta una estructura secundaria similar a la de un trébol. Tiene un brazo inferior donde se encuentra el anticodón para el reconocimiento del codón del RNA mensajero en la síntesis proteica; en la parte superior se encuentra la secuencias CCA 3' terminal, la cual es esencial para la unión del aminoácido por medio de la enzima aminoacil-tRNA sintetasa. Es importante notar que la secuencia CCA no se encuentra codificada en todos los tRNAs y que la enzima tRNA-nucleotidiltransferasa es capaz de añadir el final CCA de manera postranscripcional. Dado que actualmente se cuenta con la secuencia completa del genoma de gran número de bacterias, es importante determinar en qué genes de tRNA está presente el final CCA codificado, a fin de saber si su distribución tiene una relación evolutiva. Para ello se hizo un análisis bioinformático con 703 genomas totalmente secuenciados utilizando el programa tRNAscan-SE para predecir genes de tRNA. Se utilizó la paquetería de EMBOSS para la extracción de secuencias, tuberías de programación en UNIX para contabilizar la información y gráficos Heatmap para visualizar los datos. Se obtuvieron 40,765 secuencias de tRNAs para todos los genomas analizados, con un promedio por genoma de 58 tRNAs ± 21. De estas secuencias, 2,309 corresponden a arqueas con un promedio de 44.4 tRNAs ± 6.58 por genoma y 38,456 genes de tRNA en eubacterias con un promedio por genoma de 59.01 ± 21.68. Los phyla más conservados con respecto a la presencia de la secuencia CCA 3' terminal en la mayoría de sus tRNAs son: Betaproteobacterias (0.992 ± 0.01), Alfaproteobacterias (0.92 ± 0.186), Thermotogales (0.99 ± 0.017), Epsilonproteobacterias (0.991 ± 0.32) y Gamaproteobacterias (0.943 ± 0.158). Los phyla que presentan variabilidad en la presencia de la secuencia CCA terminal codificada son: Firmicutes (0.627 ± 0.322), Crenarchaeota (0.618 ± 0.27), Euryarchaeota (0.42 ± 0.259) y Deltaproteobacterias (0.889 ± 0.206). Finalmente, el phylum con menor presencia de la secuencia CCA es Cyanobacteria (0.288 ± 0.032). De estos datos se observa que la distribución de la secuencia CCA en el extremo 3' terminal de los tRNAs no es congruente con la la filogenia. Existe una distribución heterogénea donde los phyla ancestrales no necesariamente tiene la misma distribución que los phyla descendientes.

(Palabras clave: tRNA, CCA, edición postranscripcional, bioinformático)

SUMMARY

Transfer RNA (tRNA) consists of a single stranded chain ~ 76 nucleotides long, capable of forming a secondary structure similar to a clover. The lower arm has the anticodon sequence for codon recognition on messenger RNA during protein synthesis; at the top 3' terminus is the sequence CCA which is essential for the binding of the amino acid by the enzyme aminoacyl-tRNA synthetase. However, the CCA sequence is not encoded in all tRNAs and post-transcriptional addition by a tRNA-nucleotidyltransferase is necessary to obtain functional tRNAs. Given the large number of complete bacterial genomes it is possible to determine for each genome the distribution of tRNAs that have an encoded 3' end CCA and analyze whether there is an evolutionary relationship between the number and classes of tRNAs lacking an encoded CCA. Bioinformatics analysis were made with 703 completely sequenced genomes using the program tRNAscan-SE to predict tRNA genes; EMBOSS was used for sequence manipulation along with UNIX scripting to gather information, as well as Heatmap graphics in R to display the data. A total of 40,765 tRNAs sequences were obtained for all analyzed genomes. Each genome has an average of 21 ± 58 tRNA genes. About 2,309 tRNA genes are from archaea with an average of 44.4 ± 6.58. The grand total of eubacterial tRNA genes is 38,456 tRNAs with an average per genome of 21.68 ± 59.01. The presence of the CCA sequence is better preserved in the following order: Betaproteobacteria (0.992 ± 0.01), Alphaproteobacteria (0.92 ± 0.186), Thermotogae (0.99 ± 0.017), Epsilonproteobacteria (0.991 ± 0.32),and Gamaproteobacteria (0.943 ± 0.158). The genomes exhibiting most variation in the conesrvation of the encoded 3' CCA are: Firmicutes (0.627 ± 0.322), Crenarchaeota (0.618 ± 0.27), Euryarchaeota (0.42 ± 0.259), and Deltaproteobacteria (0.889 ± 0.206). The phylum with the least number of encoded CCA sequences at the 3' end of tRNAs is the Cyanobacteria (0.288 ± 0032). We show that there is no congruence between phylogenetic distribution and encoded CCA conservancy at the 3' end of tRNAs. Moreover, here are shown clear examples of inconsistencies between ancestor-descendant in the conservancy of the encoded CCA.

(Key words: tRNA, CCA, post-transcriptional edition, bioinformatics)

AGRADECIMIENTOS

A mi Padre, por la educación que me dio, confianza y apoyo.

A mi Madre, por su apoyo y por creer siempre en mí.

A mi familia por su apoyo continúo.

Al Dr. Juan Campos Guillén, por su ayuda, apoyo, tiempo, dedicación e invaluable amistad. Por aceptarme en su laboratorio, corregir mis errores y dirigir mi camino en la ciencia.

A la Dra. Gabriela Olmedo Álvarez, por su ayuda y apoyo. Por aceptarme en su laboratorio.

A Luis David Alcaraz Peraza, por ser un amigo y un tutor. Por instruirme en las novedosas técnicas.

A Varinia López Ramírez por su ayuda, paciencia y amistad.

A mis profesores, por instruirme en el campo de la biología.

A mis amigos: Axini, Anaid, Aurelio, Mario, Alejandra, Alfonso, Sebastián, Patricia, Gerardo, Óscar, Karina, Horacio, Marco, Gustavo, Diana y Janet. Por su ayuda y apoyo cada vez que los he necesitado, y por compartir una época de mi vida.

A mis compañeros del Lab. de Microbiología de la FCN: Laura, Claudia, Marycruz y Osvaldo por su paciencia, ayuda y amistad.

Al 9° Verano de la Ciencia Región Centro por darme la oportunidad de hacer una estancia de investigación en el Lab. de la Dra. Gabriela Olmedo Álvarez.

A la SEP por la beca otorgada para la finalización de la tesis.

A la beca otorgada FOMIX QRO. 2008 CO2-102052.

Al Apoyo SNI-Estudiantes otorgado por CONACYT para el desarrollo del proyecto (solicitud 000000000102740).

INDICE

Pagina

INDICE DE FIGURAS

Pagina

INDICE DE TABLAS

Pagina

I. INTRODUCCIÓN

El RNA de transferencia (tRNA) fue propuesto por Francis Crick como adaptador entre el RNA mensajero (mRNA) y la síntesis proteica. Fue comprobado experimentalmente por Zamecnik y Hoagland encontrando que es una molécula esencial en la síntesis proteica y dependiente de guanosina trifosfato (Hoagland *et al.*, 1958; Kresge et al., 2005). Es una cadena de RNA de pequeña longitud (73 a 93 nucleótidos), encargada de transferir un aminoácido específico a la cadena polipeptídica creciente en el sitio aminoacil del ribosoma durante la traducción de mRNA a proteína. El tRNA contiene la secuencia CCA en el sitio 3' terminal, que es esencial para la unión del aminoácido mediante la enzima aminoacil-tRNA sintetasa. El enlace entre el tRNA y el aminoácido es de tipo covalente. Una secuencia esencial para la correcta traducción de mRNA a proteína es el anticodón situado en el brazo inferior del tRNA, que consiste en tres nucleótidos complementarios al codón, código genético empleado por el mRNA. Si bien cada tipo de molécula del tRNA puede unirse específicamente a un sólo un tipo de aminoácido, el código genético contiene codones múltiples específicos para el mismo aminoácido, por lo que tRNAs que llevan anticodones diferentes pueden llevar el mismo aminoácido (Bailly *et al*, 2006; Clark, 2006; McClain, 2006; Saks y Conery, 2007).

Todos los tRNAs tienen características comunes de secuencia y estructura que les permite interactuar con las diferentes enzimas implicadas en la traducción, pero a la vez cada uno de los tRNAs posee características particulares que los dotan de especificidad. La estructura general de los tRNAs está dada de acuerdo a la disposición y naturaleza de nucleótidos, los cuales son numerados desde el extremo 5' al 3' según la secuencia estándar más común de los tRNAs que consta de 76 bases (Dieter y Uttam, 1995; Lewin, 1997; Clark, 2006).

Una característica universal de las moléculas de tRNA es la presencia de la secuencia CCA en el extremo 3' terminal. Sin embargo, no todos los tRNAs tienen la secuencia CCA codificada en el extremo 3' por lo que ésta se adiciona postranscripcionalmente, lo cual es esencial para la adición del

aminoácido. Es posible que los primeros tRNAs tuvieran siempre la terminación CCA final como parte de la secuencia. No es claro en qué grupos evolutivos se prescinde de esta secuencia para ser adicionada postrancripcionalmente. Por esta razón es importante el análisis de presencia-ausencia de la secuencia CCA codificada como parte del tRNA, su relación con el anticodón y la relación evolutiva con los grupos bacterianos. Una forma de hacer estos análisis es por medios bioinformáticos que es una disciplina científica emergente que utiliza tecnología informática para organizar, analizar y distribuir información biológica con la finalidad de responder preguntas complejas en biología, principalmente de biología molecular.

II. REVISION DE LITERATURA

RNA de transferencia

El ácido ribonucleico de transferencia (tRNA) estuvo entre los primeros ácidos nucleicos secuenciados, debido a su pequeño tamaño, y porque es posible purificar tRNAs individuales. Fue secuenciado en 1965 por Robert Holley del grupo de la Universidad de Cornell, Nueva York. La secuencia reveló una característica inesperada, que los nucleótidos que conforman el tRNAs contienen una serie de nucleótidos modificados, que van de 5 a 10 en un tRNA, y con más de 50 diferentes modificaciones conocidas en conjunto (Holley *et al.*, 1965; Brown, 2002; Clark, 2006).

La primer secuencia de tRNA examinada fue tRNA-Ala de *Saccharomyces cerevisiae*, donde la secuencia mostró que la molécula podría adoptar diversas formas dependiendo de los pares de bases en su estructura secundaria (Holley *et al.*, 1965). Después de haber secuenciado más tRNAs se hizo evidente que una estructura particular podría ser adoptada por todos ellos, tomando una forma de trébol, como se muestra en la figura 1, con las siguientes características: un brazo aceptor formado por siete pares de bases entre el 5' y 3' extremos de la molécula. Los aminoácidos se adjuntan al extremo 3' final del tRNA, en la secuencia CCA terminal. El brazo D, así llamado por el nucleósido modificado dihidrouridina, que está siempre presente en esta estructura. El brazo del anticodón que contiene el triplete de nucleótidos denominado anticodón que es la base par con que se aparea el codón del mRNA durante la traducción. El bucle variable V que contiene 3-5 nucleótidos en la clase 1 de los tRNAs o 13-21 nucleótidos en la clase 2 de los tRNAs. El brazo TψC, nombrado por la secuencia de timidina-pseudouridina-citosina, que siempre está presente. La estructura de trébol puede presentarse prácticamente en todos los tRNAs, excepto principalmente en los tRNAs mitocondriales de vertebrados, que son codificados por el genoma mitocondrial y que a veces carecen de partes de la estructura. Un ejemplo es el tRNASer mitocondrial del humano, que no tiene el brazo D en la estructura del tRNA además de carecer de la estructura secundaria conservada (Rossmanith,

1997), donde las identidades de los nucleótidos en algunas posiciones son completamente invariables (siempre el mismo nucleótido) o semi-invariables (siempre una purina o una pirimidina), y las posiciones de los nucleótidos modificados son casi siempre los mismos (Brown, 2002).

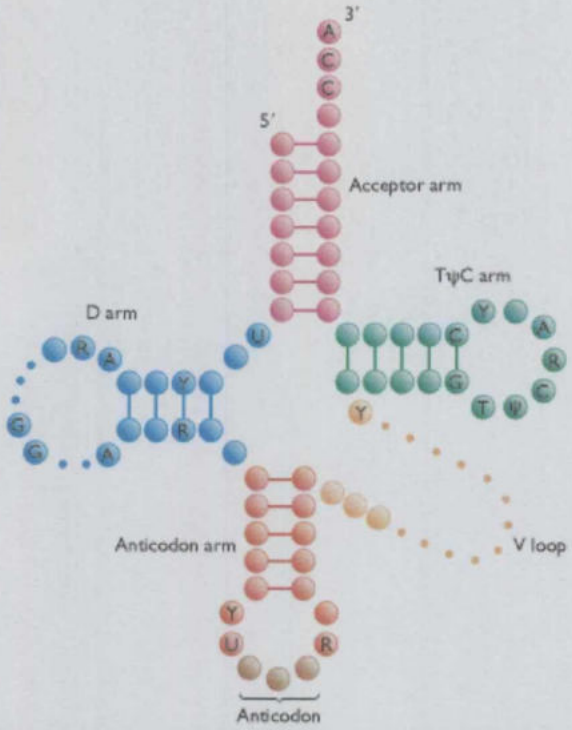

Figura 1. Estructura convencional del tRNA en forma de trébol, donde se señalan nucleótidos invariables (A, C, G, T, U, Y, donde ψ es pseudouridina) y nucleótidos -variables (con las siglas: R, purina; Y, pirimidina). Nucleótidos facultativo que no esté presente en todos los tRNAs se muestran como pequeños puntos. El sistema de numeración estándar es del extremo 5' al extremo 3´. Los nucleótidos del anticodón están en las posiciones 34, 35 y 36, (Brown, 2002) de acuerdo con la convención de Cold Spring Harbor tRNA Meeting en 1979 (Steinberg *et al*, 1993).

Estudios de cristalografía de rayos X han demostrado que los nucleótidos en el brazo D y bucles TψC forman pares de bases que doblan al tRNA en un conjunto apareado compacto que le da una estructura con una forma de L (Clark, 2006). El aumento de bases apareadas significa que la base de apilamiento es casi continua al tRNA proporcionando estabilidad a la estructura en (Brown, 2002). Basados en las características del tRNA y los nucleótidos invariables de posición se han creado programas computacionales

que buscan estas características en secuencias genómicas completas. Tal es el caso del programa tRNAscan-SE, que identifica el 99-100% del tRNA de genes en la secuencia del DNA, mientras que presenta un error de menos de un falso positivo por cada 15 gigabases analizadas. El programa detecta tRNA inusuales homólogos como selenocisteina de tRNAs, derivados de elementos repetitivos y pseudogenes de tRNA (Lowe y Eddy, 1997).

Código genético

El código genético, consiste en la codificación de los nucleótidos en tripletes para los codones del mRNA para la síntesis proteica, teniendo una codificación única y general entre todos los organismos como se muestra en la figura 2. Existen variaciones en la codificación de codones dentro de grupos taxonómicos y entre los organelos celulares eucariotas. Estas particularidades fueron observadas por Frederick Sanger en 1979 con los genomas mitocondriales que tienen variaciones particulares en la codificación de los codones para los tRNAs (Brown, 2002).

UUU phe	UCU ser	UAU tyr	UGU cys
UUC phe	UCC ser	UAC tyr	UGC cys
UUA leu	UCA ser	UAA stop	UGA stop
UUG leu	UCG ser	UAG stop	UGG trp
CUU leu	CCU pro	CAU his	CGU arg
CUC leu	CCC pro	CAC his	CGC arg
CUA leu	CCA pro	CAA gln	CGA arg
CUG leu	CCG pro	CAG gln	CGG arg
AUU ile	ACU thr	AAU asn	AGU ser
AUC ile	ACC thr	AAC asn	AGC ser
AUA ile	ACA thr	AAA lys	AGA arg
AUG met	ACG thr	AAG lys	AGG arg
GUU val	GCU ala	GAU asp	GGU gly
GUC val	GCC ala	GAC asp	GGC gly
GUA val	GCA ala	GAA glu	GGA gly
GUG val	GCG ala	GAG glu	GGG gly

Figura 2. Código genético general, utilizado en la interacción codón-anticodón.

Aminoacil-tRNA sintetasa

La aminoacil-tRNA sintetasa (AARSs) es uno de los principales componentes en el mecanismo de traducción a proteína. Estas enzimas son esenciales; se encuentran en todas las formas de vida y son responsables de

la adición correcta de los aminoácidos a los extremos CCA 3' de los tRNAs durante la síntesis proteica. La evolución de las tRNA sintetasas es de fundamental importancia en relación con la naturaleza biológica y la transición de un mundo de RNA al mundo moderno, dominado por proteínas-enzimas (O'Donoghue y Luthey-Schulten, 2003). Hay AARSs específicas para cada uno de los 20 aminoácidos estándar. Estas enzimas se dividen en dos clases, la clase I y clase II, las cuales no están relacionados tanto en secuencia y estructura. La clase I de AARSs son específicas para 11 aminoácidos, que son Met, Val, Ile, Leu, Cys, Glu, Gln, Arg, Trp, Tyr y Lys. La clase II de sintetasas especifican 10 aminoácidos, Ala, His, Pro, Thr, Ser, Gly, Phe, Asp, Asn y Lys. Sólo para el aminoácido Lys existen AARSs en las dos clases de proteínas, la AARS-Lys clase I se encuentra en la mayoría de las Archaea y en algunas eubacterias, mientras que la AARS-Lys clase II se encuentra en todos los genomas eucariotas, la mayoría de las eubacterias, y un pequeño numero de Archaea. Tanto la clase I como la clase II de proteinas AARS coexisten en dos organismos de Arqueales del genero *Methanosarcina*, M. *barkeri* y *M. acetivorans* (O'Donoghue y Luthey-Schulten, 2003).

Secuencia CCA y enzimas relacionadas

Todos los tRNAs maduros tienen la secuencia CCA en su extremo 3' terminal para ser funcionales. Esta secuencia es el sitio para la adición de aminoácidos a través de la enzima aminoacil-tRNA-sintetasa que se requiere durante la síntesis de proteínas (Cooper, 2000).

La secuencia CCA terminal está codificado en el DNA de algunos genes de tRNA, pero en otros está ausente, y en este caso se requiere una etapa de procesamiento del RNA donde enzimas reconocen el extremo 3' terminal carente de la secuencia CCA y añaden dicha secuencia a estos tRNAs. La maduración de los tRNA en su extremo 3' terminal es un proceso complicado en las bacterias. Por lo general, es iniciado por las endonucleasas RNasa E y Z en diferentes bacterias. En *Escherichia coli*, la RNasa E se une a sitios ricos en AU de las secuencias de tRNA, produciendo la transformación intermedia con

un par de residuos en el extremo 3', que luego son eliminadas por el recorte de exoribonucleasas para generar la madurez del extremo 3' (Li *et al*, 2005).

El actual modelo de procesamiento del tRNA sugiere que la RNasa E lleva a cabo la primera etapa de maduración del tRNA. Se ha demostrado que la RNasa E se adhiere a una serie de transcritos primarios del tRNA de unos pocos nucleótidos río abajo del extremo 3'. En una segunda etapa la RNasa P genera la madurez del extremo 5' final. Los pocos residuos extra en el extremo 3' se eliminan por exoribonucleasas, principalmente RNasa T y PH, simultáneamente, después de la acción de la RNasa P. Otras exoribonucleasas, como las RNases D y BN, y la inespecífica RNasa II, retiran los residuos incorrectos del extremo 3' o de lo contrario se ve disminuido el crecimiento celular (Li *et al*, 2005).

La ribonucleasa P (RNasa P), es una enzima esencial que cataliza la maduración del extremo 5' de tRNAs en todos los reinos de vida (Hartmann y Hartmann, 2003). Este tipo de ribonucleasa se encuentra en todas las bacterias con pocas excepciones (Li *et al*, 2005).

La holoenzima bacteriana RNasa P procesa el extremo 5' de prácticamente todos los tRNAs. Consiste en una subunidad de RNA de aproximadamente 400 nucleótidos y una pequeña proteína básica de ~13 kDa (Wegscheid y Hartmann, 2007). En arqueas y eubacterias se a demostrado que in vitro, las subunidades de RNA de la RNasa P son catalíticamente activas en la ausencia de la subunidad proteica (Hartmann y Hartmann, 2003).

La RNasa P bacteriana reconoce principalmente el tallo aceptor y el brazo T del precursor del tRNA (ptRNA) (Chen, 1998), mientras que la región del tallo del anticodón, el brazo D y el brazo variable no se ha elucidado su función de reconocimiento (Nagai *et al.*, 2003).

Por otra parte, la adición de la secuencia CCA en el extremo 3' de los tRNAs se ve mediada por enzimas que construyen o reparan dicha secuencia. En algunos organismos, por enzimas como la CCA-nucleotidiltransferasa la cual es capaz de adicionar la secuencia CCA en el extremo 3' de los tRNAs carentes de ella, esto sin que la enzima utilice un templado o secuencia molde

para su adición (Tomita y Weiner., 2001; Tomari *et al.*, 2002; Neuenfeldt *et al.*, 2008). Existen distintos organismos que requieren de dos tipos de enzimas para la adición de la secuencia CCA esto lo hacen en forma escalonada la primera enzima añade un CC y la segunda añade una A (Tomita y Weiner., 2001; Neuenfeldt *et al.*, 2008).

Las enzimas CCA-nucleotidiltransferasa y sus variantes evolutivas (Neuenfeldt *et al.*, 2008) utilizan un sistema de evaluación de un nucleótido a la vez y emplea como sustrato citidina trifosfato (CTP) y adenosina trifosfato (ATP) (Tomita y Weiner, 2001).

Las enzimas que adicionan la secuencia CCA son polimerasas especializadas en agregar en un orden específico la secuencia C-C-A en los extremos 3' de los tRNAs carentes de la misma secuencia. Las enzimas que adicionan la secuencia CC y A tienen relaciones evolutivas con la enzima CCA-nucleotidiltransferasa. Estas enzimas carecen de un fragmento de la secuencia de aminoácidos con respecto a la de la enzima que adiciona el CCA y por esta característica la enzima sólo adiciona la secuencia CC a los tRNAs carentes de la secuencia CCA. En estudios experimentales a la CC-nucleotidiltransferasa se le adicionaron los aminoácidos carentes respecto a la CCA-nucleotidiltransferasa y observaron que la enzima modificada adiciona la secuencia CCA a los tRNAs carentes de ella (Neuenfeldt *et al.*, 2008).

Análisis filogenéticos de las CC-nucleotidiltransferasas indican que estas enzimas emergieron varias veces durante la evolución, considerándose que descienden de la CCA-nucleotidiltransferasa donde existe una adición completa de la secuencia CCA (Neuenfeldt *et al.*, 2008).

Se a demostrado a través de enfoques genéticos y bioquímicos que la adición de la secuencia CCA en los tRNAS de *Aquifex aeolicus* requiere de la colaboración de dos enzimas una que adicione un CC y otra que añada una A como proceso final de edición (Tomita y Weiner, 2001).

III. JUSTIFICACIÓN

Debido a que se conoce que la secuencia final CCA en el extremo 3' de los tRNAs es esencial para la interacción con la enzima aminoacil-tRNAsintetasa para la unión del aminoácido, y que esta no siempre está codificada, es importante conocer qué secuencias de tRNA cuentan con la secuencia CCA terminal codificada, y cuales no, y analizar si hay relación con los anticodones y relación evolutiva con grupos bacterianos. Si las secuencias de tRNA carecen de la secuencia CCA terminal debe haber un mecanismo enzimático involucrado que añada esta secuencia.

IV. OBJETIVOS

Objetivo general

Analizar la presencia y ausencia de extremos CCA 3' codificados en los tRNAs de genomas de eubacterias y de arqueas completamente secuenciados a fin de saber si su distribución tiene una relación evolutiva.

Objetivos específicos

Obtener una base de datos que contenga sólo secuencias de tRNA de eubacterias y de arqueas.

Obtener el uso de anticodones de tRNA bacteriano respecto a su anticodón y su aminoácido codificado.

Obtener los tres últimos nucleótidos de las secuencias de los tRNAs bacterianos del extremo 3'.

Obtener la relación de los anticodones con la presencia-ausencia de los finales de secuencia CCA en el extremo 3' de los tRNAs bacterianos.

V. METODOLOGIA

Descarga de datos iniciales

Para el análisis bioinformático de secuencias de tRNAs bacterianos se empleo la base de datos disponible en GenBank del Nacional Center for Biotechnology Information (Benson *et al*, 2008) disponibles al 10 de julio de 2008, de donde se extrajeron secuencias de genomas completos correspondientes a DNA Bacteriano de la dirección electrónica: ftp://ftp.ncbi.nih.gov/genomes/Bacteria/all.fna.tar.gz la cual contiene 705 genomas, de las cuales 52 genomas corresponden a arqueas y 653 genomas corresponden a eubacterias, como se muestra en la tabla 1 de distribución de genomas empleados ordenados por Phylum.

Tabla 1. Distribución de los genomas por Phylum

Phylum	# de genomas
Crenarchaeota	17
Nanoarchaeota	1
Euryarchaeota	34
Acidobacteria	2
Actinobacteria	53
Alphaproteobacteria	88
Aquificae	2
Bacteroidetes/Chlorobi	22
Betaproteobacteria	56
Chlamydiae/Verrucomicrobia	13
Chloroflexi	7
Cyanobacteria	33
Deinococcus-Thermus	4
Deltaproteobacteria	19
Epsilonproteobacteria	18
Firmicutes	145
Fusobacteria	1
Gammaproteobacteria	165
Other Bacteria	6
Planctomycetes	1
Spirochaetes	11
Thermotogae	7

Base de datos de tRNAs de arqueas y eubacterias

De la base de datos del GenBank los archivos se organizaron en dos grupos los de arqueas y las eubacterias. Empleando los comandos **mv** para mover las carpetas de acuerdo con la clasificación taxonómica del archivo de la base de datos de la dirección electrónica: ftp://ftp.ncbi.nih.gov/genomes/Bacteria/lproks_1.txt y los archivos fueron encadenados por arqueas y eubacterias con el comando **cat** con la condición ***.fna**.

Posteriormente se hizo un filtrado de las secuencias obtenidas para obtener las coordenadas de secuencias, predicción de anticodones y predicción de transporte de aminoácidos de tRNAs, de los archivos que contiene los genomas, empleando el programa de predicción de estructuras de tRNA, tRNAscan-SE el cual identifica 99-100% de los genes de tRNAs en la secuencia de DNA mientras que menos da un falso positivo por cada 15 gigabases analizadas (Lowe y Eddy, 1997). Los parámetros usados para el análisis de predicion de tRNAs son como se muestran en los ejemplos. Para el caso de arqueas los parámetros fueron los siguientes:

```
tRNAscan-SE v.1.23 (April 2002) - scan sequences for transfer RNAs

  Please cite:
      Lowe, T.M. & Eddy, S.R. (1997) "tRNAscan-SE: A program for
      improved detection of transfer RNA genes in genomic sequence"
      Nucl. Acids Res. 25: 955-964.

  This program uses a modified, optimized version of tRNAscan v1.3
  (Fichant & Burks, J. Mol. Biol. 1991, 220: 659-671),
  a new implementation of a multistep weight matrix algorithm
  for identification of eukaryotic tRNA promoter regions
  (Pavesi et al., Nucl. Acids Res. 1994, 22: 1247-1256),
  as well as the RNA covariance analysis package Cove v.2.4.2
  (Eddy & Durbin, Nucl. Acids Res. 1994, 22: 2079-2088).

-------------------------------------------------------------
Sequence file(s) to search:  ArchivoArchaea.fna
Search Mode:                 Archaeal
Results written to:          salidaArchaea.out
Output format:               Tabular
Searching with:              tRNAscan + EufindtRNA -> Cove
Covariance model:            TRNA2-bact.cm
tRNAscan parameters:         Strict
EufindtRNA parameters:       Relaxed (Int Cutoff= -36)
tRNA secondary structure
    predictions saved to:    salidaArchaea.str
```

Por otro lado, para el caso de eubacterias los parámetros empleados fueron:

```
tRNAscan-SE v.1.23 (April 2002) - scan sequences for transfer RNAs

  Please cite:
      Lowe, T.M. & Eddy, S.R. (1997) "tRNAscan-SE: A program for
      improved detection of transfer RNA genes in genomic sequence"
      Nucl. Acids Res. 25: 955-964.

  This program uses a modified, optimized version of tRNAscan v1.3
  (Fichant & Burks, J. Mol. Biol. 1991, 220: 659-671),
  a new implementation of a multistep weight matrix algorithm
  for identification of eukaryotic tRNA promoter regions
  (Pavesi et al., Nucl. Acids Res. 1994, 22: 1247-1256),
  as well as the RNA covariance analysis package Cove v.2.4.2
  (Eddy & Durbin, Nucl. Acids Res. 1994, 22: 2079-2088).

--------------------------------------------------------------------
Sequence file(s) to search:  ArchivoBacteria.fna
Search Mode:                 Bacterial
Results written to:          salidaBacteria.out
Output format:               Tabular
Searching with:              tRNAscan + EufindtRNA -> Cove
Covariance model:            TRNA2-bact.cm
tRNAscan parameters:         Strict
EufindtRNA parameters:       Relaxed (Int Cutoff= -36)
tRNA secondary structure
    predictions saved to:    salidaBacteria.str
```

De la predicción de genes de tRNA con el programa tRNAscan-SE se obtuvieron dos archivos uno es la salida de resultados donde presenta a modo de columnas el nombre de la secuencia de entrada, numero de tRNA en la secuencia analizada, coordenadas, aminoácido al que corresponde, anticodón, coordenadas de intrones y score de la predicción de la secuencia. Como se muestra en la salida del programa a modo de ejemplo:

Sequence		tRNA	Bounds	tRNA	Anti	Intron	Bounds	Cove
Name	tRNA #	Begin	End	Type	Codon	Begin	End	Score
---	---	---	---	---	---	---	---	---
NC_009925	1	29248	29320	Arg	TCT	0	0	82.00
NC_009925	2	118983	119056	Arg	ACG	0	0	64.84
NC_009925	3	290954	291028	Asp	GTC	0	0	76.28

Del segundo archivo de salida del tRNAscan-SE corresponde a la predicción de estructura secundaria. Donde en la primer línea indica el nombre de la secuencia, el numero del tRNA dentro de la secuencia analizada, las coordenadas dentro de la secuencia y la longitud en pares de bases (pb); en la segunda línea, el tipo de aminoácido correspondiente, el anticodón, las coordenadas del anticodon, y la calificación de la predicción; en la cuarta línea la secuencia correspondiente al tRNA; en la quinta línea, se representan los apareamientos internos de los nucleótidos con un signo > ó < y los puntos representan nucleótidos no apareados, de esa forma representa la estructura secundaria del tRNA como en el siguiente ejemplo:

```
NC_009925.trna1 (29248-29320)    Length: 73 bp
Type: Arg    Anticodon: TCT at 34-36 (34-36)  Score: 82.00
         *    |    *    |    *    |    *    |    *    |    *    |    *    |
Seq: GCGCTCGTAGCTCAGCGGAtAGAGCAGTTGCCTTCTAAGCAATTGGtCGCAGGTTCGAGTCCTGCCGAGCGCG
Str: >>>>>>>..>>>>.......<<<<.>>>>>.......<<<<<.....>>>>>.......<<<<<<<<<<<.

NC_009925.trna2 (118983-119056)  Length: 74 bp
Type: Arg    Anticodon: ACG at 35-37 (35-37)  Score: 64.84
         *    |    *    |    *    |    *    |    *    |    *    |    *    |
Seq:
GGGCTTGTAGCTCAGTGGActAGAGCACGTGGCTACGGACCACGGTGtCGGGGGTTCGAATCCCTCCTAGCCCG
Str:
>>>>>.>..>>>>.........<<<<.>>>>>.......<<<<<.....>>>>>.......<<<<<<.<<<<<.

NC_009925.trna3 (290954-291028)  Length: 75 bp
Type: Asp    Anticodon: GTC at 36-38 (36-38)  Score: 76.28
         *    |    *    |    *    |    *    |    *    |    *    |    *    |
Seq:
GGGGGTGTAGCTCAGTCGGTctAGAGCAGTCGCCTGTCGAGCGAAAGGtCGCGGGTTCAAATCCCGTCTCCCCCG
Str:
>>>>>.>..>>>>..........<<<<..>>>>.......<<<<......>>>>>.......<<<<<<.<<<<<.
```

De las coordenadas de secuencias de tRNA obtenidas con el programa tRNAscan-SE se compilaron en un script en BASH donde se llama a la paquetería *seqret* de la suite EMBOSS (Rice *et al*, 2000) como se muestra en el siguiente ejemplo.

```
seqret Bacteria.fna: NC_009925 -auto -stdout -sbegin 29248 -send 29320
```

De donde se obtuvieron secuencias en formato FASTA (Pearson y Lipman, 1988). Como en el siguiente ejemplo:

```
>NC_009925
GCGCTCGTAGCTCAGCGGAtAGAGCAGTTGCCTTCTAAGCAATTGGtCGCAGGTTCGAGTCCTGCCGAGCGCG
```

Teniendo así una base de datos con secuencias de genes correspondiente a tRNAs de aqueas y eubacterias.

Uso de anticodones y aminoácidos en los tRNAs

Para el uso de anticodones y aminoácidos se combino la salida de resultados del tRNAsca-SE con la salida del programa EMBOSS, de la siguiente forma:

De la salida del tRNAscan-SE se utilizaron los parámetros de salida tRNA#, tipo de tRNA (corresponde al aminoácido) y anticodón; de la salida del programa EMBOSS se remplazo la etiqueta de cada secuencia (>NC_009925) reestructurando la etiqueta del siguiente modo: ">"Identificador de la secuencia"-tr"(#del tRNA en el genoma)"-"aminoácido correspondiente"-"anticodón del tRNA. Como se muestra en el siguiente ejemplo.

```
>NC_009925-trl-Arg-TCT
GCGCTCGTAGCTCAGCGGAtAGAGCAGTTGCCTTCTAAGCAATTGGtCGCAGGTTCGAGTCCTGCCGAGCGCG
```

Obtención de los tres últimos nucleótidos de cada secuencia

Partiendo de la base de datos con los identificadores modificados que contiene información de identificador del genoma, numero de tRNA en el genoma, aminoácido y anticodón correspondiente a cada tRNA. Se ejecutó el script en perl get_contig_ends.pl (ver Apéndice) modificado del sitio http://www.genome.ou.edu/informatics.html obteniendo los tres últimos nucleótidos de cada secuencia de tRNA en las secuencias analizadas, y algunos finales de secuencia fueron corregidos de forma manual. Finalmente, los tres últimos nucleótidos de la secuencia fueron adicionados a las etiquetas de sus correspondientes secuencias de tRNA como se muestra en el siguiente ejemplo:

```
>NC_009925-trl-Arg-TCT_GCG
GCGCTCGTAGCTCAGCGGAtAGAGCAGTTGCCTTCTAAGCAATTGGtCGCAGGTTCGAGTCCTGCCGAGCGCG
```

Organización de la información

Para organizar la información de las secuencias y etiquetas, se edito la información a modo de tabla. Como se describe a continuación: se remplazaron los identificadores en las etiquetas por los nombres correspondientes a las cepas como para el caso del identificador NC_009925 por la primera letra del

dominio al cual corresponde (A para arquea y B para eubacteria) seguido del nombre Acaryochloris_marina_MBIC11017, empleando un script en BASH usando el comando **sed** de forma global para los remplazos, quedando las secuencias como en el siguiente ejemplo:

```
>BAcaryochloris_marina_MBIC11017-tR1B-Arg-TCT_GCG
GCGCTCGTAGCTCAGCGGAtAGAGCAGTTGCCTTCTAAGCAATTGGtCGCAGGTTCGAGTCCTGCCGAGCGCG
```

Después se filtraron las etiquetas de las secuencias con el comando **grep** con los parámetros **-i '>'** para quedarnos con las filas de las etiquetas, después se filtraron las etiquetas de acuerdo al nombre de la cepa generando archivos individuales a los cuales se les contabilizo el numero de secuencias para cada uno de los 20 aminoácidos, para los aminoácidos modificados, el numero de secuencias para cada uno de los 64 tripletes del código genético, con un script en BASH empleando el comando **grep –i –c**. (ver Apéndice) las salidas de los conteos se ordenaron en forma de tabulador en fila para poder encadenar los conteos con el comando **cat** y así obtener una matriz de datos manipulable en hoja de calculo. Ya con los datos se procedió a estandarizarlos usando la formula # de tRNA con el CCA codificado en el extremo 3' / total de tRNAs. Teniendo así valores contemplados de cero a uno, y con valores de NA donde no se contara con ningún tRNA para ese parámetro.

Relación del aminoácido con la presencia ausencia de CCA 3'

De datos estandarizados se seleccionaron los que correspondían a los 20 aminoácidos estándar con el respectivo nombre de cada cepa. Estos se emplearon para hacer una gráfica tipo Heat Maps empleando el código modificado de Susko (2006:http://www.mathstat.dal.ca/~tsusko ver Apéndice) en el programa estadístico R. Este tipo de gráficos representa datos en dos dimensiones de modo de un mapa de calor donde los colores del mapa representan valores de una matriz de datos a modo de coordenadas X, Y o píxeles. Con esta herramienta es posible mostrar una gran cantidad de datos en un gráfico, además de tener la posibilidad de un análisis de conglomerados tanto en el eje X como en el Y. Este tipo de gráficos se emplean normalmente en biología molecular, para representar el nivel de expresión de muchos genes a través de una serie de muestras comparables como en los micro arreglos.

Relación del anticodón con la presencia-ausencia de CCA 3'

De los datos estandarizados se utilizo la información de los genomas correspondiente a los 64 anticodones, y la codificación de antidodones se transformo a codones de acuerdo al código genético estándar. Se contabilizaron los genomas que presentaban tRNA para cada codón y se contabilizaron los genomas que presentan la secuencia CCA 3' terminal en los tRNAs. Estos datos se representaron en un grafico en forma circular de acuerdo con el código genético y a los aminoácidos correspondientes para cada codón (Figura 6).

VI. RESULTADOS Y DISCUSIÓN

Se obtuvieron 40765 secuencias de tRNAs para todos los genomas analizados con un promedio por genoma de 58 tRNAs ± 21. De éstos, 2309 corresponden a arqueas con un promedio de 44.4 tRNAs ±6.58 y 38456 a eubacterias con un promedio por genoma de 59.01 tRNAs ± 21.68.

La distribución de las especies respecto del agrupamiento de datos considerando la presencia-ausencia de la secuencia CCA 3' terminal en relación con el aminoácido correspondiente para ser transportado se observa en la Figura 3, donde los colores claros al blanco indican presencia de CCA codificado y los tonos al rojo indican la ausencia de CCA codificado. Puede observarse que no hay una distribución de las especies de acuerdo con la filogenia de éstas comparado con el árbol filogenetico reportado por Ciccarelli (2006), en la figura 4. Sin embargo, se encuentran pequeños grupos taxonómicos de bacterias que tienen todos o casi todos los tRNAs con el extremo CCA 3' terminal codificado. Tal es el caso de algunas α, β, δ, ε-proteobacteria y algunos Firmicutes de los cuales destacan todos lo *Mycoplasma*, y después las γ-proteobacterias, sin embargo estos grupos no están completos encontrando especies de estos grupos a lo largo de la gráfica. Las cepas de la especies *E. coli* se encuentran en la en la zona media del grupo de las γ-proteobacterias intercaladas con *Salmonella* y *Xanthomonas* y no se asocian a un grupo especifico. El genero *Bacillus* se encuentra en la parte media del gráfico por no tener la secuencia CCA 3' codificada en la mayoría de sus tRNAs. Dado que el número de cepas analizadas son muchas, la resolución necesaria en este gráfico para poder apreciar los nombres y los datos no sería suficiente en este formato, por lo que se indican los mayores grupos en la Figura 3.

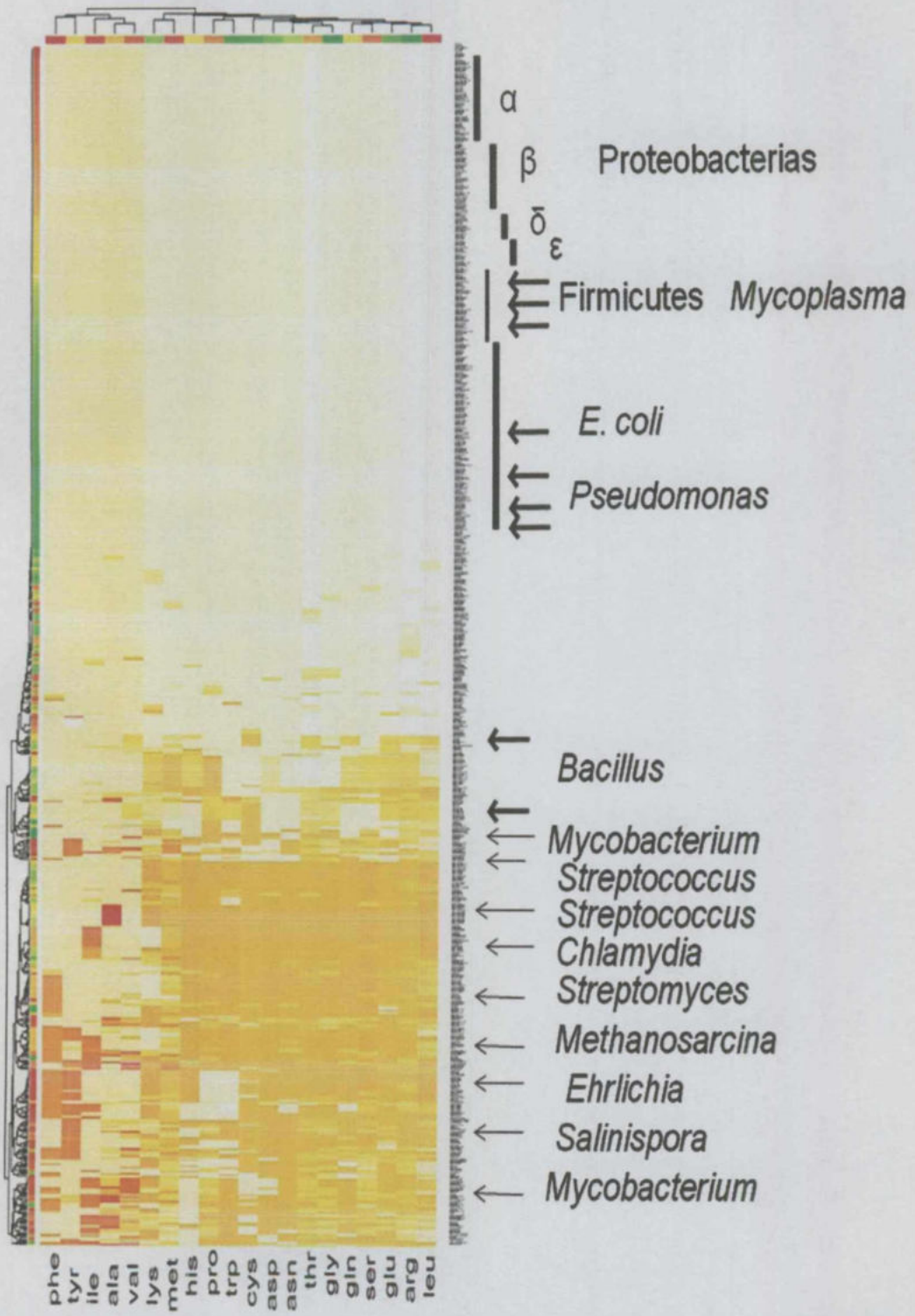

Figura 3. Heat Map de las cepas de eubacterias y de arqueas en base a la presencia ausencia de la secuencia CCA 3' terminal en sus tRNAs respecto al aminoácido, en colores claros al blanco presencia de la secuencia CCA y en colores cercanos al rojo ausencia de la secuencias.

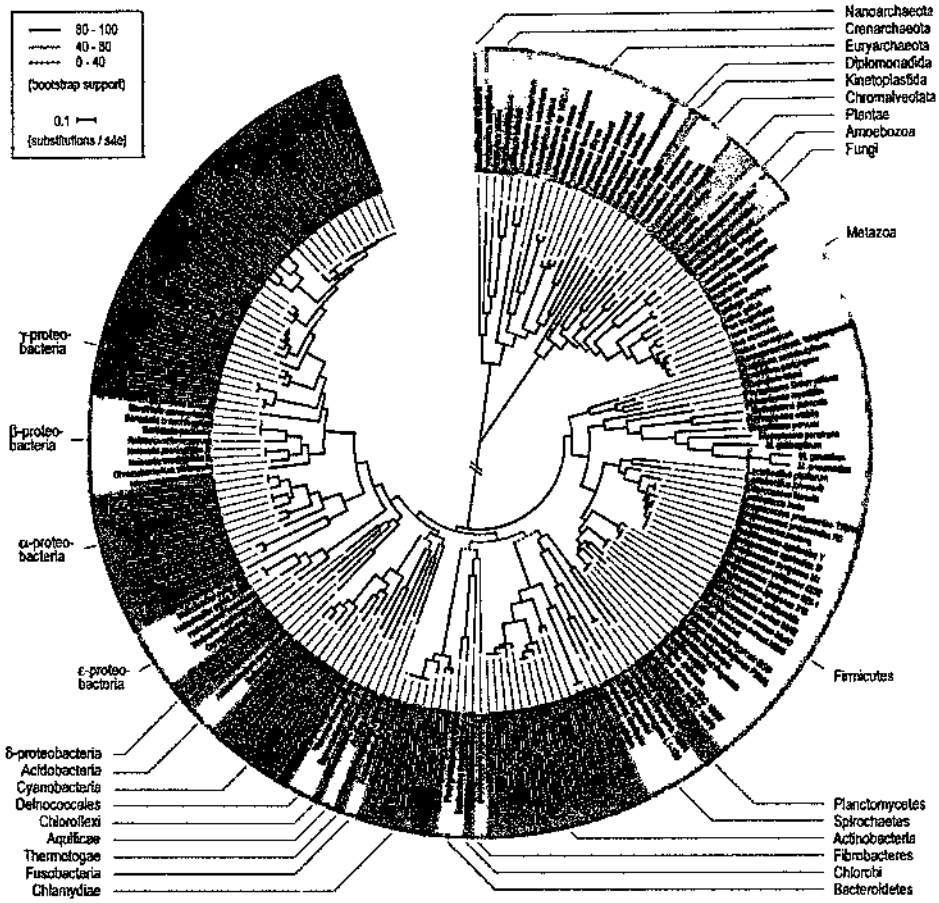

Figura 4. Filogenia global de organismos totalmente secuenciados, basado en 31 proteínas presentes en todos los organismos, considerando 191 especies, donde en verde corresponde a arqueas, en rojo eucariontes y en azul a eubacterias (Tomada de Ciccarelli, 2006).

Para sintetizar la información, se decidió emplear los grandes grupos filogenéticos descritos por Ciccarelli (2006), mismas que se aprecian en la figura 5 donde se muestran las especies ordenadas por la presencia-ausencia de la secuencia CCA en el extremo 3' terminal de los tRNAs de acuerdo con el aminoácido que codifican. También se observan pequeños grupos como los de *Mycoplasma*, *Fusobacterium*, *Pseudomonas* y *Vibrio* que contienen todos sus tRNAs codificados con la secuencia 3' terminal. Mientras que los grupos de *Salmonella, E. coli* y *Xantomonas* al menos uno de sus representantes carecen de secuencias CCA en el extremo 3', mientras que para el género

Bacillus, el 0.7437 ± 0.039 de sus tRNAs presentan la secuencia CCA. Para los grupos *Chlamydia, Methanosarcina* y *Mycobacterium* se observa un bajo contenido de CCA codificado en el extremo 3' terminal.

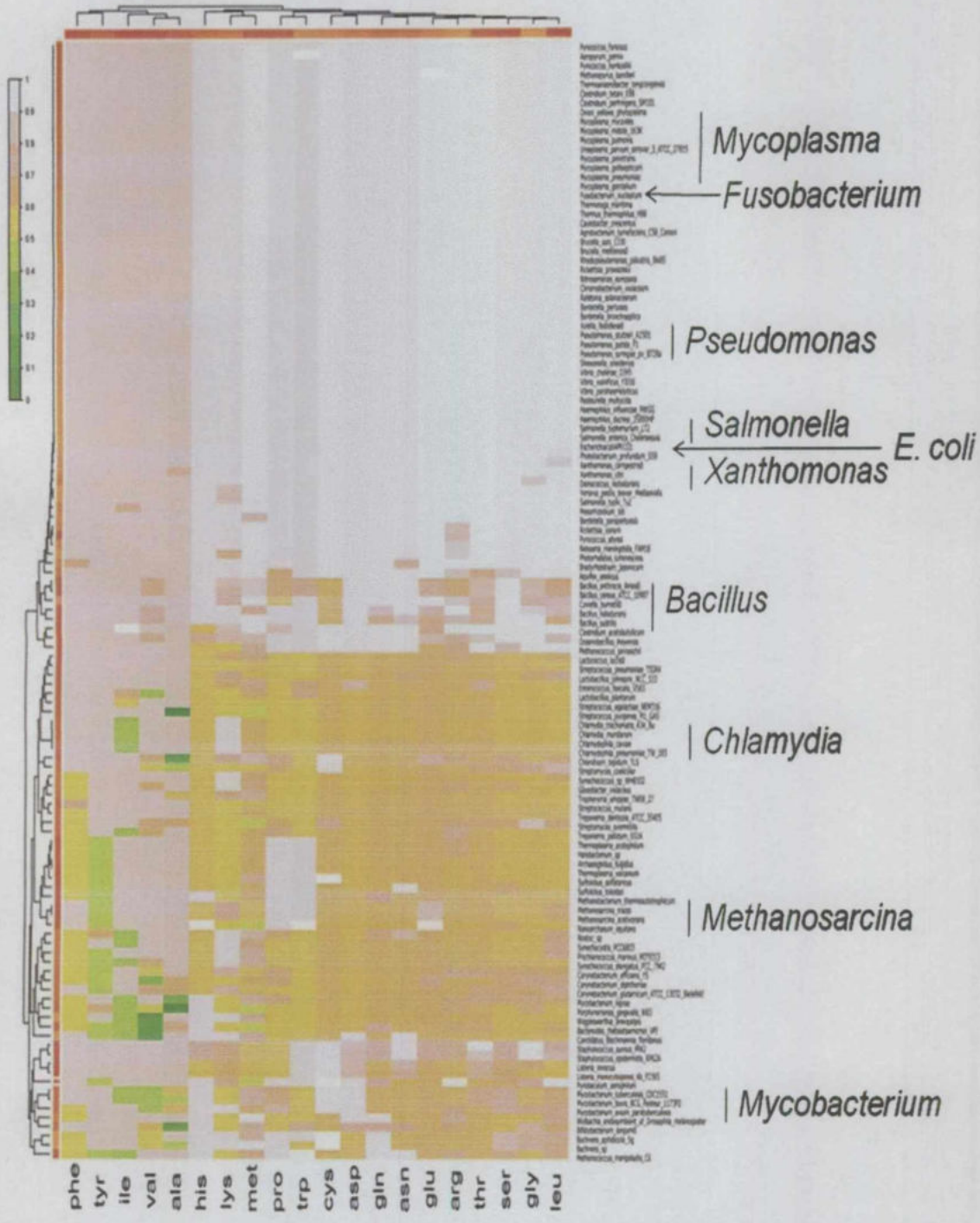

Figura 5. Heat map de las cepas bacterianas y de arqueas en base a la presencia ausencia de la secuencia CCA 3' terminal en sus tRNAs respecto al aminoácido considerando las especies empleadas por Ciccarelli en 2006 para

construir su árbol filogenético, los colores rosa al blanco indican la presencia de la secuencia CCA' y los tonos verde indican la ausencia de la secuencia.

Por otra parte, para el conteo de anticodones y la traducción a codones se graficó la presencia-ausencia de la secuencia CCA 3' terminal en los tRNAs en un esquema de circulo que se muestra en la Figura 6.

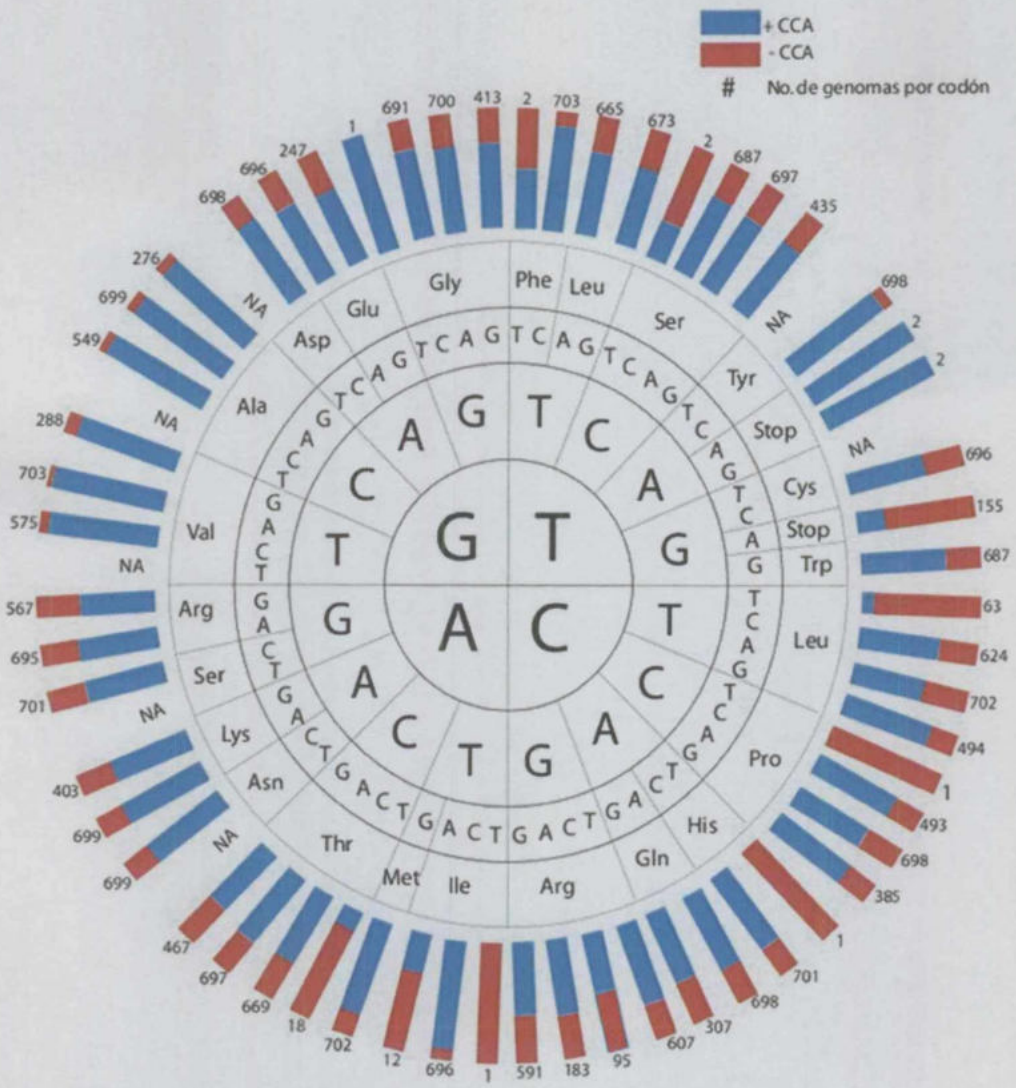

Figura 6. Presencia y distribución de tRNAs de 703 genomas de arqueas y eubacterias por codón con respecto a la presencia de la secuencia CCA 3' terminal. En la cual se puede ver en color azul se representa la presencia de CCA en el extremo 3' terminal y en color rojo la ausencia de dicha secuencia, y el número del extremo es la cantidad genomas que contienen secuencias para ese codón.

Para la síntesis de proteínas, el ribosoma requiere dos tipos de tRNAs: de iniciación y elongación. El de iniciación con el codón ATG correspondiente al anticodón CAT del tRNA para formilmetionina en procariotas y de metionina especies eucariontes (Ivanov *et al.*, 2001; Simonetti *et al.*, 2009). Donde para los datos obtenidos se tiene que 702 genomas contiene el tRNA con anticodón CAT para metionina, de los cuales 0.7568 ± 0.3318 tienen la secuencia CCA codificada en le extremo 3' terminal no encontrándose ninguna relación de la presencia de la secuencia CCA con respecto a los grupos taxonómico.

Mientras que para la elongación comprende a los 61 codones para tRNAs respecto al código genético que codifican para los 20 aminoácidos estándar (Linn *et al.*, 2002). En nuestro análisis, obtuvimos 54 codones que codifican para los 20 aminoácidos estándar y uno para un aminoácido modificado a selenocysteina.

Del codón de CGC para arginina que corresponde al anticodón GCG del tRNA, 94 genomas lo presentan con características particulares; está presente en la mayoría de las arqueas, la mitad de las Spirochaetas, algunos *Mycoplasma* y todos los Thermotogae, Epsilonproteobacteria y Chloroflexi, estos últimos tienen el extremo 3' terminal codificado con CCA. Con respecto a este codón, Lobry y Necşulea (2006) reportan que es empleado por organismos termófilos e hipertermófilos. Sin embargo, nuestros datos no muestran evidencia contundente de preferencia por dichos microorganismos. El codón CGT para arginina, que corresponde al tRNA con anticodón ACG, lo presentan 607 genomas de lo cual destaca la ausencia de los grupos: Archaea, Thermotogae, Epsilonproteobacteria, Chloroflexi y la mitad de las Spirochaetas. Mientras que de los organismos que si presentan este tRNA, los que tienen el CCA 3' codificado tienen la siguientes características: El CCA está presente encasi todos los tRNA Arg (CGT) de Betaproteobacteria (1 ± 0), Deinococcus-Thermus (1 ± 0), Gammaproteobacteria (0.940 ± 0.221), Alphaproteobacteria (0.909 ± 0.289). Muestran gran variación: Deltaproteobacteria (0.894 ± 0.315), Aquificae (0.5 ± 0.707), Firmicutes (0.471 ± 0.449), Actinobacteria (0.1 ± 0.278), Bacteroidetes/Chlorobi (0.221 ± 0.398), Actinobacteria (0.1 ± 0.278); Está ausente en Cyanobacteria (0 ± 0), Chlamydiae/Verrucomicrobia (0 ± 0). Lobry y Necşulea (2006) mencionan que este codón es empleado de forma

mínima por bacterias termófilas, lo cual concuerda con los datos obtenidos además de que la mayoría de las eubacterias extremófilas y ninguna arquea usa este codón.

Lobry y Necşulea (2006) del mismo modo reportan que los codones AGA (Arg), ATA (Ile) y AGG (Arg) son empleados por organismos termófilos. En nuestros resultados no observamos la preferencia de los termófilos por estos codones, además de que son usados por la mayoría de los organismos.

Singer y Hickey (2003) proponen que los 11 codones de mayor frecuencia para procariotas termófilos son: GGA, AGG, AGA, AAG, AAC, ATA, TAC, TTC, CAC, CTT y CTC. Comparando con nuestros datos estos codones corresponden a un uso generalizado de los procariontes sin importar su tolerancia a temperatura. Aunque los organismos que prefieren el codón CTT son algunas especies de los géneros: *Prochlorococcus*, *Streptococcus*, *Lactobacillus* y algunos otros Firmicutes. Mientras que los organismos que no usan el codón CTT por lo general usan el codón GAG que corresponde al mismo aminoácido (Leu).

Para el termino de la síntesis proteica se requieren factores de terminación que son necesarios para reconocer los codones de paro (UAG, UGA, y UAA) (Ivanov, 2001) y los tRNAs de paro no están codificados en el DNA, sin embargo el código genético varía en una amplia gama de organismos, algunos de los cuales no comparten similitudes obvias. A veces se repite el mismo cambio en diferentes linajes, tal es el caso de los codones de paro. Los codones han sido reasignados de parar la síntesis proteica a transportar un aminoácido. Del mismo modo, los animales y levaduras tienen mitocondrias independientes donde el tRNA que corresponde al codón AUA es reasignado para transporte de los aminoácidos Ile o Met (Knight, 2001). Para nuestro caso los datos obtenidos para el codón UGA correspondiente al anticodón TCA de los tRNAs, lo encontramos en los géneros: *Mycobacterium*, *Solibacter*, *Rubrobacter*, *Kineococcus*, *Aquifex*, *Burkholderia*, *Chloroflexus*, *Myxococcus*, *Desulfococcus*, *Campylobacter*, *Mycoplasma*, *Clostridium*, *Ureaplasma*, *Heliobacterium*, *Mesoplasma*, *Shigella*, *Yersinia*, *Haemophilus*, *Pseudomonas*, *Salmonella*, *Yersinia*, *Klebsiella* y algunas cepas de la especie

E. coli. Diversos reportes para este codón demuestran que *Mycoplasma capricolum* (Jukes, 1985; Yamao *et al*, 1985) y *Mycoplasma pneumoniae* (Simoneau, 1993) que utilizan el anticodón TCA con una codificación para triptófano en su código genético. Mientras que Lobry (2006) sugiere que todos los integrantes de los generos: *Mesoplasma*, *Mycoplasma*, *Spiroplasma* y *Ureaplasma* el codo UGA corresponde al aminoácido triptófano, y de acuerdo con los datos, estos géneros para el codón UGA tienen la secuencia CCA 3' terminal en sus tRNAs.

El mismo codón UGA en otros organismos como *E. coli* codifica para aminoácidos modificados como selenocisteina SelC como reporta Ambrogelly (2007) en una revisión sobre el código genético. De lo que corresponde a nuestros datos estos tRNAs carecen de la secuencia CCA en el extremo 3' terminal. Además de que las especies de los géneros *Burkholderia*, *Chloroflexus*, *Geobacter*, *Clostridium*, *Shigella*, *Yersinia*, *Shewanella*, *Salmonella*, *Klebsiella*, *Treponema* y *Pseudomonas* presenta el mismo comportamiento respecto a los datos del codón UGA en cuanto a la presencia de la secuencia CCA 3' terminal

Por otra parte, estudios con *Salmonella typhimurium* mostraron que la reconstrucción del tRNA para el codón UGA le confiere la característica de supresor (Elliott y Wang, 1991).

Del mismo modo, la distribución de la secuencia CCA 3' terminal en los tRNAs a nivel de Phylum es muy variable como se puede observar en la tabla 2. Donde se muestra el promedio para cada Phylum con su respectiva desviación estándar.

Tabla 2. Distribución del CCA 3' terminal por Phylum

Phylum	promedio	Desviacion estándar
Crenarchaeota	0,618	0,270
Nanoarchaeota	0,342	NA
Euryarchaeota	0,420	0,259
Acidobacteria	0,970	0,012
Actinobacteria	0,358	0,083
Alphaproteobacteria	0,920	0,186
Aquificae	0,595	0,347
Bacteroidetes/Chlorobi	0,258	0,199
Betaproteobacteria	0,992	0,010
Chlamydiae/Verrucomicrobia	0,211	0,023
Chloroflexi	0,963	0,035
Cyanobacteria	0,288	0,039
Deinococcus-Thermus	0,989	0,012
Deltaproteobacteria	0,884	0,206
Epsilonproteobacteria	0,991	0,032
Firmicutes	0,627	0,322
Fusobacteria	1	NA
Gammaproteobacteria	0,943	0,158
Other Bacteria	0,899	0,238
Planctomycetes	0,157	NA
Spirochaetes	0,307	0,150
Thermotogae	0,990	0,017

Donde las Betaproteobacterias son las más cohesivas en la distribución del CCA codificado al final de los tRNAs. De los 56 genomas analizados para este grupo, el valor menor de frecuencia de CCA codificado es de 0.96, pero la mayoría de los genomas analizados presentan tRNAs con CCA codificado en un rango de 0.99 a 1 con un promedio de 0.992 ± 0.01. Las bacterias pertenecientes al Phylum Thermotogae presentan un comportamiento similar con un promedio de 0.990 ± 0.0.17 de CCA codificado en los apenas siete genomas. Mientras que las Cyanobacterias son el Phylum donde abundan los tRNAs que carecen del CCA codificado, y la frecuencia de tRNAs sin CCA fluctúa desde 0.2 a 0.35 en diferentes genomas analizados con un promedio de 0.289 ± 0.039. El Phylum más interesante es el de Firmicutes, donde algunos géneros, como *Mycoplasma* y *Clostridium*, poseen el mayor número de genomas cuyos tRNAs poseen tRNAs con el CCA codificado, mientras que genomas de los de los géneros *Lactobacillus* y *Streptococcus* tienen

frecuencias que fluctúan entre 0.18 y 0.36. Los *Bacillus* y *Staphilococcus* tienen valores intermedios, entre 0.63 y 0.8. Ningún otro Phylum tiene una distribución tan amplia en los valores de presencia/ausencia del CCA 3' terminal codificado como el de los Firmicutes con un promedio de 0.627 ± 0.322. Prácticamente todos los datos son congruentes dentro de cada género, y la excepción más notable es en los *Clostridium*, donde un solo un genoma (*Clostridium phytofermentans*) presenta un valor de 0.18 en la frecuencia de tRNAs que codifican el CCA, que contrasta enormemente con los de los otros genomas, que están arriba de 0.9. En realidad esta especie de *Clostridium* se aleja bastante de las otras especies secuenciadas de *Clostridium*, que corresponden a patógenos.

En general, pareciera haber una restricción a que el CCA terminal codificado se pierda en betaproteobacterias, mientras que otros Phylum como Firmicutes, Deltaproteobacteria y Aquificae, parecen ser más relajados en este sentido.

VII. CONCLUSIONES

La distribución de la secuencia CCA en el extremo 3' terminal de los tRNAs de manera global no es de acuerdo a la filogenia, dando como consecuencia una distribución heterogénea donde los phyla ancestrales no necesariamente tiene la misma distribución que los phyla descendientes.

Considerando la distribución de la secuencia CCA 3' terminal en los diferentes phyla podría ser una pérdida de esta secuencia por phylum ancestral aunado por la ganancia de mecanismos de maduración del tRNA. Mientras que los phyla recientes, conservan más las secuencias CCA.

VIII. PERSPECTIVAS

Seria conveniente analizar a detalle el phylum Firmicutes por su alta variabilidad en la distribución del CCA, demás de las enzimas involucradas en el proceso de adición de la secuencia CCA además de las enzimas implicadas en la maduración de los tRNAs.

De igual manera, sería interesante comparar las enzimas participantes en edición y maduración de los tRNAs de los phyla más conservado como lo son: Betaproteobacterias, Alfaproteobacteria, Epsilonproteobacteria y Gamaproteobacteria. Contra los phyla que presentan variabilidad en la presencia CCA codificada como son: Firmicutes, Crenarchaeota, Euryarchaeota, Euryarchaeota y Deltaproteobacteria

Por otro lado, seria favorable estudiar a detalle los genomas que presentan baja codificación de la secuencia CCA 3' terminal en los tRNAs con el fin de analizar las enzimas involucradas en la edición del CCA si como en el proceso de maduración de los tRNAs esto en los phyla que presentan mayor codificación de la secuencia CCA como el caso del phylum Betaproteobacteria, Thermotogae y Gammaproteobacteria.

IX. LITERATURA CITADA

Ambrogelly A., Palioura S. y Söll D. (2007). Natural expansion of the genetic code. *Nature Chemical Biology*. **3**(1) 29-35.

Bailly, M., Giannouli, S., Blaise, S., Stathopoulos, C., Kern, D. y Becker, H. D. (2006). A single tRNA base pair mediates bacterial tRNA-dependent biosynthesis of asparagines. *Nucleic Acids Research*. **34**(21): 6083–6094.

Benson D., Karsch-Mizrachi I., Lipman D., Ostell J. y Wheeler D. (2008). GenBank. *Nucleic Acids Research*, **36**, D25–D30.

Brown, T.A. 2002 Genomes 2ª ed. New York and London: Garland Science.

Chen J., Nolan J., Harris M. y Pase N. (1998). Comparative photocross-linking analysis of the tertiary structures of *Escherichia coli* and *Bacillus subtilis* RNase P RNAs. *EMBO Journal*. **17**(5) 1515-1525.

Ciccarelli F., Doerks T., Mering C., Creevey C., Snel B. y Bork P. (2006). Toward Automatic Reconstruction of a Highly Resolved Tree of Life. *Science*. **311**(3) 1283-1287.

Clark, B. F. (2006). The crystal structure of tRNA. *J. Biosci*. **31**(4): 453–457.

Cooper, Geoffrey M. 2000. The Cell - A Molecular Approach. 2ª ed. Sunderland (MA): Sinauer Associates, Inc.

Dieter, S. y Uttam, L. R. 1995. tRNA Structure, biosynthesis, and Function. ASM Press. Washington, D. C.

Elliott T. y Wang X. (1991). Salmonella typhimurium prfA Mutants Defective in Release Factor 1. *J Bacteriol*. **173**(13) 4144-4154.

Hartmann E. y Hartmann R. K. (2003). The enigma of ribonuclease P Evolution. *Trends in Genetics* **19**(10) 561-659.

Hoagland M., Stephenson M., Scott J., Hecht L. y Zamecnik P. (1958) A soluble ribonucleic acid intermediate in protein synthesis. *J Biol Chem.* **231**(1):241-57.

Holley, R. W., J. Apgar, G. A. Everett, J. T. Madison, M. Marquisee, S. H. Merrill, J. R. Penswick, y A. Zamir. (1965). Structure of a ribonucleic acid. *Science* **147**:1462-1465.

Ivanov V., Beniaminov A., Mikheyev A. y Minyat E. (2001). A mechanism for stop codon recognition by the ribosome: A bioinformatic approach. *RNA.* **7**: 1683-1692.

Jukes T. H. (1985). A Change in the Genetic Code in *Mycoplasma capricolum. J Mol Evol.* **22**: 361-362.

Knight R., Freeland S. y Landweber L. (2001). Rewiring the keyboard: evolvability of the genetic code. *Nature Genetics Rev.* **2**:40-58.

Kresge N., Simoni R. y Hill R. (2005). The Discovery of tRNA by Paul C. Zamecnik. *JBC.* **280**(40)e37-e39.

Lewin, B. 1997. Genes 6°ed. Oxford University Press, EUA.

Li Z., Gong X., Joshi V. y Li M. (2005). Co-evolution of tRNA 3' trailer sequences with 3' processing enzymes in bacteria. *RNA*, **11**:567–577.

Linn D., Singer G. y Hickel D. (2002). Synonymous codon usage is subject to selection in thermophilic bacteria. *Nucleic Acids Research.* **30**(19) 4272-4277.

Lobry J. y Neçsulea A. (2006). Synonymous codon usage and its potential link with optimal growth temperature in prokaryotes. *Gene.* **385**: 128-136.

Lowe, T. M. y Eddy, S. R. (1997). tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Research*, **25**(5) 955–964.

McClain, W. H. (2006). Surprising contribution to aminoacylation and translation of non-Watson–Crick pairs in tRNA. *PNAS*. 103(12) 4570-4575.

Nagai Y., Ando T., Tanaka T. y Kikuchi. (2003). Recognition of tRNA bottom half by bacterial ribonuclease P. *Nucleic Acids Research Supplement* **3**:281-282.

Neuenfeldt A., Just A., Betat H. y Mörl M. (2008). Evolution of tRNA nucleotidyltransferases: A small deletion generated CC-adding enzymes. *PNAS*. **105**(23) 7953-7958.

O'Donoghue P. y Luthey-Schulten Z. (2003). On the Evolution of Structure in Aminoacyl-tRNA Synthetases. *Microbiology and molecular biology reviews*. **67**(4) 550–573.

Pearson WR, y Lipman DJ. (1988). Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*. **85** (8):2444-2448.

Rice, P., Longden, I., Bleasby, (2000). "EMBOSS: the European Molecular Biology Open Software Suite.", *Trends in Genet.*, **16** (6) 276–277.

Rossmanith W. (1997). Processing of human mitochondrial tRNAGCUSer(AGY): a novel pathway in tRNA biosynthesis. *Journal of Molecular Biology*. **265**(4):365-371.

Saks, M. E. y Conery, J. S. 2007. Anticodon-dependent conservation of bacterial tRNA gene sequences. *RNA*. **13**: 651-660.

Simoneaau P., Li C., Loechel S. Wenzel R. Herrmann R. y Hu P. (1993). Codon reading scheme in *Mycoplasma pneumonia* revealed by the analysis of the complete set of tRNA genes *Nucleic Acids Research*. **21**(21)4967-4974.

Simonetti A., Marzi S., Jenner L., Myasnikiv A., Romby P., Yusupova G., Klaholz P. y Yusupov M. (2009). A structural view of translation initiation in bacteria. *Cell. Mol. Life Sci.* **66**: 423-436.

Singer G. A. C. y Hickey D. A. (2003). Thermophilic prokaryotes have characteristic patterns of codon usage, amino acid composition and nucleotide content. *Gene*. **317**: 39-47.

Steinberg S, Misch A y Sprinzl M. 1993. Compilation of tRNA sequences and sequences of tRNA genes. *Nucleic Acids Res*. **21**(13):3011-3015.

Susko E., Leigh J., Doolittle W. F. y Bapteste E. (2006). Visualizing and Assessing Phylogenetic Congruence of Core Gene Sets: A Case Study of the *g*-Proteobacteria *Mol. Biol. Evol*. **23**(5)1019–1030.

Tomari Y., Suzuki T. y Ueda T. (2002). tRNA Recognition by CCA-adding enzyme. *Nucleic Acids Research Supplement*. **2**: *77-78*

Tomita K. y Weiner A. M. (2001). Collaboration Between CC- and A-Adding Enzymes to Build and Repair the 3*-Terminal CCA of tRNA in *Aquifex aeolicus*. *Sciece*. **294**: 1334-1336.

Wegscheid B. y Hartmann R. 2007. In vivo and in vitro investigation of bacterial type B RNase P interaction with tRNA 3' -CCA. *Nucleic Acids Res*. **35**(6): 2060–2073.

Yamao F., Muto A. Kawauchi Y., Iwami M., Iwagami S., Azumi Y. y Osawa S. (1985). UGA is read as tryptophan in Mycoplasma capricolum. *Proc. Nat. Acad. Sci. USA*. **82**: 2306-2309.

APÉNDICE

Script get_contig_ends.pl

```
#!/usr/local/bin/perl -w
$Date_Last_Modified = "April 10, 2006";
$DEFAULT_LINE_LENGTH = 80;    # Default length of fasta output sequence
lines.
$output_line_len = $DEFAULT_LINE_LENGTH;# Length of fasta output
sequence lines.
$MIN_LINE_LENGTH = 10;        # Mimimum allowed value for
$output_line_len.
$MIN_CONTIG_LENGTH = 1;       # Minimum contig length.
$min_contig_length = $MIN_CONTIG_LENGTH; # Default minimum contig
length.
                        #   Shorter contigs are discarded.  (See -c)
$end_length = 0;        # Default contig end length.
$shorten_contig_name = 0;     # Is contig name shortened? (See -s)
$preserve_comments = 0;       # Are contig header comments preserved?
(See -p)
$get_qualities = 0;           # Is a fasta quality file to be
used/created?
$reverse_and_complement = 0;  # Are output contigs to be reversed and
                        #   complemented?
$Verbose_Mode = 0;            # Are some statistics listed? (see -v)
$strip_X_N = 0;               # Are leading and trailing Xs and Ns
removed?
                        #   (See -x)
$strip_Z0 = 0;                # Are leading and triling 0 quality bases
                        #   removed (See -z)
$join_ends = 0;               # Join the ends into one contig (See -j)
$join_char = '';        #   Character to be duplicated for join string
$join_len = 50;               # Length of string to join contig ends
#$join_string = "$join_char" x $join_len;  # String for joining contig
ends
#@JOIN_QUALS = ('0') x $join_len;  # Dummy qualities for joined string
$fill_join = 0;               # Join the ends into one contig and fill
the
                        #   middle with fill characters to preserve
                        #   the original length (See -f)
$directory_separator = '/';   # For Unix
#$directory_separator = '\';  # For DOS/Windows
$full_command_name = $0;
if ($full_command_name =~
m"(^|${directory_separator})([^${directory_separator}]*)$")
  {
  $my_name = $2;
  }
else
  {
  $my_name = 'get_contig_ends';
  }

while ($ARGV[0] =~ /^-/)
  {
  $flag = shift @ARGV;
  $f = substr($flag, 1, 1);
  $value = $flag;
  substr($value, 0, 2) = '';

  if ($f eq 'e')        # -e end_length
```

```
    {
    if ($value ne '')
      {
      $end_length = $value;
      }
    else
      {
      $end_length = shift @ARGV;
      }
    if ($end_length !~ /^\d+$/)
      {
      display_help("Invalid 'end_length': -e $end_length");
      exit 2;
      }
    }
  elsif ($f eq 'c')           # -c min_contig_length
    {
    if ($value ne '')
      {
      $min_contig_length = $value;
      }
    else
      {
      $min_contig_length = shift @ARGV;
      }
    if ($min_contig_length !~ /^\d+[KMGkmg]?$/)
      {
      display_help("Invalid 'min_contig_length': -c
$min_contig_length");
      exit 2;
      }
    $min_contig_length =~ s/k$/000/i;
    $min_contig_length =~ s/m$/000000/i;
    $min_contig_length =~ s/g$/000000000/i;
    $min_contig_length =~ s/^0+//i;
    $min_contig_length = '0' if $min_contig_length eq '';
    if ($min_contig_length < $MIN_CONTIG_LENGTH)
      {
      display_help("Invalid 'min_contig_length': -c
$min_contig_length");
      exit 2;
      }
    }
  elsif ($f eq 'f')     # -f fill_char    (replace middle of contig
with fill
    {             #            characters and middle quals with 0s)
    $fill_join = 1;
    if ($value ne '')
      {
      $join_char = "$value";
      }
    else
      {
      $join_char = shift @ARGV;
      }
    }
  elsif ($f eq 'l')           # -l output_line_length
    {
    if ($value ne '')
      {
      $output_line_len = $value;
```

```
      }
    else
      {
      $output_line_len = shift @ARGV;
      }
    if (($output_line_len !~ /^\d+$/) || $output_line_len <
$MIN_LINE_LENGTH)
      {
      display_help("Invalid 'output_line_length': -c
$output_line_len");
      exit 2;
      }
    }
  elsif ($f eq 'j')           # -j join_char    (join contig ends into
    {                    #            single contigs)
    $join_ends = 1;
    if ($value ne '')
      {
      $join_char = "$value";
      }
    else
      {
      $join_char = shift @ARGV;
      }
    $join_string = "$join_char" x $join_len;  # String for joining
contig ends
    @JOIN_QUALS = ('0') x $join_len;
    }
  elsif ($f eq 's')           # -s  (shorten contig names)
    {
    $shorten_contig_name = 1;
    }
  elsif ($f eq 'p')           # -p  (preserve contig header comments)
    {
    $preserve_comments = 1;
    }
  elsif ($f eq 'q')           # -q  (also extract fasta quality file)
    {
    $get_qualities = 1;
    }
  elsif ($f eq 'r')           # -r  (reverse and complement output
contigs)
    {
    $reverse_and_complement = 1;
    }
  elsif ($f eq 'v')           # -v  (verbose mode)
    {
    $Verbose_Mode = 1;
    }
  elsif ($f eq 'x')           # -x  (remove leading and trailing Xs and
Ns)
    {
    $strip_X_N = 1;
    }
  elsif ($f eq 'z')           # -z  (remove leading and trailing zero
quals)
    {
    $strip_Z0 = 1;
    }
  elsif ($f eq 'h')           # -h  (help)
    {
```

```
    display_more_help();
    exit 0;
    }
  else
    {
    display_help("Invalid flag: $flag");
    exit 2;
    }
  }
  if ($fill_join && $join_ends)
    {
    display_help("Cannot specify both -f and -j");
    exit 2;
    }
  $single_contig_length = 2 * $end_length;
  $single_contig_length += length($join_string) if ($join_ends ||
$fill_join);
  $strip_Z0 &= $get_qualities;      # Can't strip zero quals without
scores
$fasta_input_file = shift @ARGV;
$fasta_input_file = '-' if (!$fasta_input_file);
if ($get_qualities && $fasta_input_file eq '-')
  {
  display_help("Missing 'fasta_input_file' name (and
'fasta_output_file' name),\n  which are required when '-q' is
specified.");
  exit 2;
  }
open(FASTAIN, $fasta_input_file) || die("Can't open fasta_input_file:
'$fasta_input_file'\n");
my $qualin = '';
if ($get_qualities)
  {
  if (-f "${fasta_input_file}.qual")
    {
    $qualin = "${fasta_input_file}.qual";
    if (!open(QUALIN, "$qualin"))
      {
      close(FASTAIN);
      die("Can't open input fasta quality file: '$qualin'\n");
      }
    }
  else
    {
    $qualin = $fasta_input_file;
    unless ($qualin =~ s/\.f(ast|n)?a$/.qual/ && -f $qualin)
      {
      close(FASTAIN);
      die("Can't find input fasta quality file for
'$fasta_input_file'\n");
      }
    if (!open(QUALIN, $qualin))
      {
      close(FASTAIN);
      die("Can't open input fasta quality file: '$qualin'\n");
      }
    }
  $Qline = <QUALIN>;
  $Qline_num = 1;
  }
$fasta_output_file = shift @ARGV;
```

```
$fasta_output_file = '-' if (!$fasta_output_file);
if ($get_qualities && $fasta_output_file eq '-')
  {
  close(FASTAIN);
  close(QUALIN);
  display_help("Missing 'fasta_output_file' name, which is required
when '-q' is specified.");
  exit 2;
  }
if (!open(FASTAOUT, ">$fasta_output_file"))
  {
  close(FASTAIN);
  close(QUALIN) if $get_qualities;
  die("Can't create fasta_output_file: '$fasta_output_file'\n");
  }
if ($get_qualities)
  {
  if (!open(QUALOUT, ">${fasta_output_file}.qual"))
    {
    close(FASTAIN);
    close(QUALIN);
    close(FASTAOUT);
    die("Can't create output fasta quality file:
'${fasta_output_file}.qual'\n");
    }
  }
print STDERR "\n$my_name - Last Modified: $Date_Last_Modified\n\n" if
$Verbose_Mode;
$Num_Contigs = 0;
$Single_Contigs = 0;
$Double_Contigs = 0;
$Skipped_Contigs = 0;
$header = '';
$contig = '';
$sequence = '';
$line_num = 0;
$Qcontig = '';
while ($line = <FASTAIN>)
  {
  chomp $line;
  $line_num++;
  if ($line =~ /^>/)
    {
    if ($contig)  # after first input line?
      {
      if ($get_qualities && length($sequence) != @Quality)
        {
        close(FASTAIN);
        close(FASTAOUT);
        close(QUALIN);
        close(QUALOUT);
        die "Lengths of fasta sequence and quality files do not match
on\n  contig='$contig'\n";
        }
      process_contig($contig, $sequence);
      }
    $Num_Contigs++;
    $header = $line;
    if ($header =~ m/^>(\S+)(.*)$/)
      {
      $contig = $1;
```

```
      $comment = $2;
      }
    else
      {
      close(FASTAIN);
      close(FASTAOUT);
      if ($get_qualities)
        {
        close(QUALIN);
        close(QUALOUT);
        }
      die "Error: Invalid fasta_input_file format:
'$fasta_input_file'\n  Fasta input line number=$line_num\n";
      }
    $sequence = '';
    if ($get_qualities)
      {
      if (! defined $Qline)
        {
        close(FASTAIN);
        close(FASTAOUT);
        close(QUALIN);
        close(QUALOUT);
        die "Error: End of file, missing fasta_qual_input_file
contig(s): '$qualin'\n  Quality file input line number=$Qline_num,\n
Sequence contig='$contig'\n";
        }
      chomp($Qline);
      $Qheader = $Qline;
      if ($Qheader =~ m/^>(\S+)(.*)$/)
        {
        $Qcontig = $1;
        $Qcomment = $2;
        if ($contig ne $Qcontig)
          {
          close(FASTAIN);
          close(FASTAOUT);
          close(QUALIN);
          close(QUALOUT);
          die "Fasta sequence and quality files do not match on contig
header number $Num_Contigs\n  Sequence contig='$contig', Quality
contig='$Qcontig'\n";
          }
        }
      else
        {
        close(FASTAIN);
        close(FASTAOUT);
        close(QUALIN);
        close(QUALOUT);
        die "Error: Invalid fasta_qual_input_file format: '$qualin'\n
Quality file input line number=$Qline_num,\n  Qline='$Qline'\n";
        }
      $Qline = <QUALIN>;
      $Qline_num++;
      $Quality = '';
      while (defined $Qline && $Qline !~ /^>/)
        {
        chomp($Qline);
        $Quality .= ' ' . $Qline;
        $Qline = <QUALIN>;
```

```
        $Qline_num++;
        }
      $Quality =~ s/^\s+//;
      $Quality =~ s/\s+$//;
      @Quality = split(' ', $Quality);
      } # end if ($get_qualities)
                                # Remove Contig name prefix?
    $contig =~ s/^.*Contig/Contig/ if $shorten_contig_name;
    next;
    } # end if ($line =~ /^>/)

  if (!$contig)
    {
    close(FASTAIN);
    close(FASTAOUT);
    if ($get_qualities)
      {
      close(QUALIN);
      close(QUALOUT);
      }
    die "Error: Invalid fasta_input_file format:
'$fasta_input_file'\n";
    }
  $line =~ s/\s//g;
  $sequence .= $line;
  } # end while

  if ($contig)
    {
    if ($get_qualities && length($sequence) != @Quality)
      {
      close(FASTAIN);
      close(FASTAOUT);
      if ($get_qualities)
        {
        close(QUALIN);
        close(QUALOUT);
        }
      die "Lengths of fasta sequence and quality files do not match
on\n  contig='$contig'\n";
      }
    process_contig($contig, $sequence);
    }
  else
    {
    print STDERR "Error: Empty fasta_input_file:
'$fasta_input_file'\n";
    }
close(FASTAIN);
close(FASTAOUT);
if ($get_qualities)
  {
  close(QUALIN);
  close(QUALOUT);
  }
if ($Verbose_Mode)
  {
  print STDERR "$Num_Contigs contigs read\n";
  if ($end_length > 0)
    {
    print STDERR "$Single_Contigs single/short contigs written\n";
```

```
    if ($join_ends || $fill_join)
      {
      print STDERR "$Double_Contigs joined contig end pairs
written\n";
      }
    else
      {
      print STDERR "$Double_Contigs contig end pairs written\n";
      }
    }
  else
    {
    print STDERR "$Single_Contigs contigs written\n";
    }
  print STDERR "$Skipped_Contigs contigs < $min_contig_length bases
skipped\n\n";
  }
exit 0;
sub process_contig
  {
  my($contig, $sequence) = @_;
  my($len, $i);

  # Strip Ns, Xs, and zero quality bases from ends of sequence?
  if ($strip_X_N && $strip_Z0)
    {
    $len = length($sequence);
    while ($len > 0 && ((substr($sequence, $len - 1, 1) =~ /^[XxNn]$/)
                     || ($Quality[$len - 1] == 0)))
      {
      $len--;
      }
    splice(@Quality, $len);
    substr($sequence, $len) = '';
    $len = @Quality;
    $i = 0;
    while ($i < ($len - 1) && ((substr($sequence, $i, 1) =~
/^[XxNn]$/)
                            || ($Quality[$i] == 0)))
      {
      $i++;
      }
    splice(@Quality, 0, $i);
    substr($sequence, 0, $i) = '';
    }

  # Strip only Ns and Xs from both ends of sequence?
  if ($strip_X_N && !$strip_Z0)
    {
    if ($sequence =~ s/^([XxNn]+)// && $get_qualities)
                                        {
      $len = length($1);
      splice(@Quality, 0, $len);
      }
    if ($sequence =~ s/([XxNn]+)$// && $get_qualities)
      {
      $len = length($1);
      splice(@Quality, -$len);
      }
    }
```

```
 # Strip only zero quality bases from ends of sequence?
 if ($strip_Z0 && !$strip_X_N)
   {
   $len = @Quality;
   while ($len > 0 && $Quality[$len - 1] == 0)
     {
     $len--;
     }
   splice(@Quality, $len);
   substr($sequence, $len) = '';
   $len = @Quality;
   $i = 0;
   while ($i < ($len - 1) && $Quality[$i] == 0)
     {
     $i++;
     }
   splice(@Quality, 0, $i);
   substr($sequence, 0, $i) = '';
   }
 $len = length($sequence);
 if ($len < $min_contig_length)  # Now look for empty reads or short
contigs
   {
   $Skipped_Contigs++;
   }
 elsif ($end_length == 0 || $len <= $single_contig_length)  # Do we
output one file?
   {
   output_contig('>' . $contig, $sequence);
   output_contig_qual('>' . $contig, @Quality) if $get_qualities;
   $Single_Contigs++;
   }
 elsif ($fill_join)    # Output paired ends as a filled joined contig?
   {
   my $fill_len = $len - $end_length - $end_length;
   output_contig('>' . $contig, substr($sequence, 0, $end_length) .
                 ($join_char x $fill_len) .
                 substr($sequence, -$end_length));
   output_contig_qual('>' . $contig, splice(@Quality, 0,
$end_length),
                      ('0') x $fill_len, splice(@Quality, -
$end_length))
     if $get_qualities;
   $Double_Contigs++;
   }
 elsif ($join_ends)    # Output paired ends as a joined contig?
   {
   output_contig('>' . $contig, substr($sequence, 0, $end_length) .
                 $join_string . substr($sequence, -$end_length));
   output_contig_qual('>' . $contig, splice(@Quality, 0,
$end_length),
                      @JOIN_QUALS, splice(@Quality, -$end_length))
     if $get_qualities;
   $Double_Contigs++;
   }
 else                  # Otherwise, output the two ends as separate
contigs
   {
   output_contig('>' . $contig . 'r', substr($sequence, 0,
$end_length));
```

```
    output_contig('>' . $contig . 'f', substr($sequence, -
$end_length));
    if ($get_qualities)
      {
      output_contig_qual('>' . $contig . 'r', splice(@Quality, 0,
$end_length));
      output_contig_qual('>' . $contig . 'f', splice(@Quality, -
$end_length));
      }
    $Double_Contigs++;
    }
  } # end process_contig
sub output_contig
  {
  my($header, $sequence) = @_;
  my($len) = length($sequence);
  my($segment, $i);
  if ($reverse_and_complement)
    {
    $sequence = reverse($sequence);
    $sequence =~ tr/acgtACGT/tgcaTGCA/;
    $header .= '.comp';
    }
  if ($preserve_comments)
    {
    $header .= $comment;      # Add contig comments back to header?
    }
  print FASTAOUT "$header\n";
  for ($i = 0; $i < $len; $i += $output_line_len)
    {
    $segment = substr($sequence, $i, $output_line_len);
    print FASTAOUT "$segment\n";
    } # end for ($i ... )
  } # end output_contig
sub output_contig_qual
  {
  my($header, @quals) = @_;
  my($len) = scalar @quals;
  my($i, $segment);
  if ($reverse_and_complement)
    {
    @quals = reverse(@quals);
    $header .= '.comp';
    }
  if ($preserve_comments)
    {
    $header .= $Qcomment;     # Add contig comments back to header?
    }
  print QUALOUT "$header\n";
  $segment = '';
  for ($i = 0; $i < $len; $i++)
    {
    if (length($segment) >= $output_line_len)
      {
      print QUALOUT "$segment\n";
      $segment = '';
      }
    $segment .= "$quals[$i] ";
    } # end for ($i ... )
  print QUALOUT "$segment\n" if (length($segment));
  } # end output_contig_qual
```

```
sub display_help
  {
  my($msg) = @_;
  print STDERR "\n$msg\n" if $msg;
  print STDERR <<EOF;

USAGE: $my_name [-e end_length] [-c min_contig_length] [-f fill_char]
            [-j join_char] [-l output_line_length]
            [-p] [-q] [-r] [-s] [-v] [-x] [-z]
            [fasta_input_file [fasta_output_file]]
              or
       $my_name -h
```

Script en BACH para el conteo de aminoácidos y anticodones por archivo

```
grep -i -c 'phe' archivo.cnt >>cont-archivo.txt
grep -i -c 'leu' archivo.cnt >>cont-archivo.txt
grep -i -c 'ile' archivo.cnt >>cont-archivo.txt
grep -i -c 'met' archivo.cnt >>cont-archivo.txt
grep -i -c 'val' archivo.cnt >>cont-archivo.txt
grep -i -c 'ser' archivo.cnt >>cont-archivo.txt
grep -i -c 'pro' archivo.cnt >>cont-archivo.txt
grep -i -c 'thr' archivo.cnt >>cont-archivo.txt
grep -i -c 'ala' archivo.cnt >>cont-archivo.txt
grep -i -c 'tyr' archivo.cnt >>cont-archivo.txt
grep -i -c 'his' archivo.cnt >>cont-archivo.txt
grep -i -c 'gln' archivo.cnt >>cont-archivo.txt
grep -i -c 'asn' archivo.cnt >>cont-archivo.txt
grep -i -c 'lys' archivo.cnt >>cont-archivo.txt
grep -i -c 'asp' archivo.cnt >>cont-archivo.txt
grep -i -c 'glu' archivo.cnt >>cont-archivo.txt
grep -i -c 'cys' archivo.cnt >>cont-archivo.txt
grep -i -c 'trp' archivo.cnt >>cont-archivo.txt
grep -i -c 'arg' archivo.cnt >>cont-archivo.txt
grep -i -c 'gly' archivo.cnt >>cont-archivo.txt
grep -i -c 'sec' archivo.cnt >>cont-archivo.txt
grep -i -c 'pseudo' archivo.cnt >>cont-archivo.txt
grep -i -c 'SeC(p)' archivo.cnt >>cont-archivo.txt
grep -i -c 'Undet' archivo.cnt >>cont-archivo.txt
grep -i -c 'TTT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TTC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TTA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TTG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TCT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TCC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TCA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TCG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TAT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TAC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TAA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TAG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TGT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TGC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TGA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'TGG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CTT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CTC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CTA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CTG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CCT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CCC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CCA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CCG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CAT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CAC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CAA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CAG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CGT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CGC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CGA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'CGG_' archivo.cnt >>cont-archivo.txt
```

```
grep -i -c 'ATT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ATC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ATA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ATG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ACT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ACC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ACA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'ACG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AAT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AAC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AAA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AAG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AGT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AGC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AGA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'AGG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GTT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GTC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GTA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GTG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GCT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GCC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GCA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GCG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GAT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GAC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GAA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GAG_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GGT_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GGC_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GGA_' archivo.cnt >>cont-archivo.txt
grep -i -c 'GGG_' archivo.cnt >>cont-archivo.txt
grep -i -c '???_' archivo.cnt >>cont-archivo.txt
grep -i -c '_CCA' archivo.cnt >>cont-archivo.txt
grep -i -c '>' archivo.cnt >>cont-archivo.txt
sed 's/\n/\t/g' cont-archivo.txt >archivo.tb.txt
```

Codigo HeatMap para R

```
secuencia <- read.table("dist.txt")
secuencia <- read.table("dist.txt",header=TRUE)
x  <- as.matrix(secuencia)
 rc <- rainbow(nrow(x), start=0, end=.1)
 cc <- rainbow(ncol(x), start=0, end=.1)
 hv <- heatmap(x, col = terrain.colors(128), scale="column",
          RowSideColors = rc, ColSideColors = cc, margin=c(8,10),
           xlab = "Clases", ylab= "Phyllum",
           main = "Distribución de taxonómica de secuencias")

secuencia <- t(secuencia) # transpose so that gene x top
brks <- seq(0.00,100.00,10.00)
grid <- expand.grid(x=c(1:dim(secuencia)[1]), y=c(1:dim(secuencia)[2]))
grid$z <- c(secuencia)
levelplot(z~x*y, data = grid, col.regions =  terrain.colors(128),
           at = brks, xlab = "Clases",
           ylab = "Phyllum",
           colorkey = list(at = brks, labels = as.character(brks)))
```