UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

INSTITUTO DE BIOTECNOLOGIA

"ANALISIS DE LA DISTRIBUCION DE LOS SITIOS DE METILACION DAM Y DCM EN EL GENOMA COMPLETO DE ESCHERICHIA COLI Y SU POSIBLE IMPLICACION BIOLOGICA"

T E S I S

QUE PARA OBTENER EL GRADO DE

LICENCIADO EN INVESTIGACION BIOMEDICA BASICA

P R E S E N T A :

CEI ABREU GOODGER

CUERNAVACA, MORELOS JUNIO DEL 2001

AGRADECIMIENTOS

Quiero agradecer a mis papás, Alberto y Jane que si no fuera por ellos esta tesis definitivamente no existiría. Mis hermanitos, Olivia y Gavin – aunque no tan culpables – no tienen excusa alguna para no aparecer justo aquí.

Carmen, Gabriel, Guillermo y Laura tuvieron la pesadísima labor de revisar esta tesis. No sólo sobrevivieron, sino que me ayudaron enormemente en el proceso. Muchísimas gracias.

He logrado (¡por fin!) llegar a este punto gracias al apoyo de todos mis amigos. De alguna manera han aprendido a aguantarme, cada quien muy a su particular manera:

Alejandro (la ex-Cosa), Amaranta (la dulce Ami), Carlitos (el Flaco), Chocobo (aka Iván), Cleila (Cle), Greeny (Darth Greeny to you!), Ileana (Liconita), Johnny (y sus derivados), Lenin (Lorena opinaba que estaba "suculento", pero nunca se le quedó), Luis (demasiados apodos para escoger uno sólo), Oscar (Puckman), Paula (la Media-Hora, Pelúcida, *pequeña*), Sole (la devoradora de alfajores), Fito (Frito Pie), Paola (mejor lo dejo así), Pilar (Pilibili), Alina (una escuincla preciosa), Ruy (arigato!), Silvia (Silvilinda, Silviux)... gracias a todos.

Hablando de amigos y de aguantarme; la persona que más paciencia me ha tenido, que más tiempo me ha dedicado, que me ha enseñado tantas cosas en este último año y medio: Enrique. Gracias, mil gracias.

Más vale tarde que nunca. Prosigamos...

ÍNDICE

INDICE.

Sección *Página*

TABLAS Y FIGURAS

Descripción breve	***Página***

INTRODUCCIÓN

Variabilidad en el DNA

A mediados del siglo pasado Watson y Crick publicaron su modelo de la estructura del DNA (Watson y Crick, 1953a). Este primer modelo consideraba a la molécula de DNA como una doble hélice, antiparalela, con un ancho constante determinado por apareamientos específicos entre una base púrica y una pirimídica. La importancia del modelo radicó en que además de encajar perfectamente con todos los hechos que se conocían hasta el momento, como la regla de Chargaff (A=T y C=G) y los datos cristalográficos de Rosalind Franklin (Franklin y Gosling, 1953), sugirió inmediatamente el mecanismo general de la replicación de esta molécula (Watson y Crick, 1953b). Sin embargo, el DNA de los organismos puede diferir bastante del modelo original (ahora conocido como DNA–B). Por un lado, condiciones como el pH, concentración de sales y temperatura, originan en algunos casos distintos tipos de hélices: más anchos (DNA–A), girando en sentido opuesto (DNA–Z), o inclusive triples hélices (DNA–H) (Wells, 1988). Por otra parte, pequeñas variaciones en las fuerzas electrostáticas entre las bases, dependiendo de cada secuencia, pueden modificar los parámetros geométricos de las bases causando variantes de los cuatro modelos de DNA antes mencionados. Así, una secuencia puede inherentemente ser más curva o simplemente ser más flexible (Trifonov y Sussman, 1980; Crothers, et al. 1990). Otras fuentes de variación muy importantes en los ácidos nucleicos de los seres vivos son las modificaciones de sus bases nitrogenadas. Estas alteraciones ocurren después de que el polinucleótido ha sido ensamblado. Basta analizar algunas de las bases que forman los tRNAs para darse cuenta de la increíble variabilidad que puede existir (Bjork, et al. 1987). Algunos de los cambios que ocurren son: metilación, saturación de dobles ligaduras, cambio de un carbono por un nitrógeno o de un oxígeno por un átomo de azufre, adición de un grupo carbonilo o grupos orgánicos mucho más complejos. Como es de esperarse, las bases modificadas tienden a comportarse diferentemente e inclusive favorecen apareamientos no canónicos (distintos de A con T/U o C con G), de hecho, en algunos casos las bases normales pueden presentar también este tipo de comportamiento. No todas las modificaciones de bases son causadas por mecanismos celulares. Muchos son daños causados por agentes externos como

radicales libres o luz ultravioleta, o inclusive pueden ocurrir por la acción directa de moléculas de agua (Weibauer, et al. 1993). Algunos efectos que esto ocasiona son: desaminación de bases que puede convertir una base en otra (por ejemplo C en U), depurinación o depirimidinación (la pérdida de la base nitrogenada) o la formación de dímeros de timina (dos timinas contiguas unidas covalentemente). Todos estos daños tienen que ser corregidos o pueden causar mutaciones y existe una serie de mecanismos de reparación que se encargan precisamente de esto, generalmente involucrando la excisión de la parte del DNA afectado y posteriormente, la síntesis de la parte faltante (Sancar y Sancar, 1988). El tipo de modificación del DNA sobre el cual este trabajo se enfoca es la metilación: la adición de un grupo $-CH_3$ a una molécula de DNA ya sintetizada.

Metilación del DNA

Curiosamente el descubrimiento de la primera base metilada, 5-metilcitosina, ocurrió cuando la estructura y función de los ácidos nucleicos seguían siendo un misterio. En 1948, Rollin Hotchkiss la identificó cuando buscaba aminoácidos en sus preparaciones de DNA (Hotchkiss, 1948), ya que en esa época se apostaba a que las nucleoproteínas y no el DNA eran las portadoras del material genético. Los métodos de cromatografía en papel que desarrolló, aunque fallaron en su propósito original (no encontró aminoácidos en sus muestras) sirvieron como una técnica rápida para la separación de purinas y pirimidinas además de descubrir la 5-metilcitosina (m5C). Otras bases metiladas se identificaron posteriormente, incluyendo la N^4-metilcitosina (m4C) y la N^6-metiladenina (m6A), cuyos grupos metilo se encuentran unidos a un nitrógeno exocíclico (Weissbach, 1993). Estas bases modificadas pueden tener consecuencias estructurales sobre el DNA que las contiene. La m5C causa una mayor estabilidad de la doble hélice, mientras que la m6A reduce esta estabilidad (Murchie y Lilley, 1989). Por lo tanto, la presencia de m6A facilita la apertura de la doble hélice, por lo que ha sido seleccionado en orígenes de replicación (discutido más adelante). La curvatura intrínseca del DNA se ve incrementada o disminuida dependiendo de las posiciones de m5C en un fragmento de DNA (Hagerman, 1990). Esta base modificada podría favorecer la formación de regiones de triples hélices (DNA-H) y estabilizar hélices invertidas (DNA-Z) en condiciones fisiológicas (Zacharias, 1993). Muchas bacterias termofílicas contienen m4C, posiblemente para evitar utilizar m5C que es propenso a desaminarse espontáneamente con mayor facilidad a temperaturas elevadas (Ehrlich, et al.

1985).

Sistemas de Restricción y Metilación

El descubrimiento de la primera función biológica de bases metiladas ocurrió a principios de los años cincuenta, cuando se observó que ciertas cepas de bacterias inhibían el crecimiento de fagos previamente crecidos en cepas diferentes (Luria y Human, 1952; Bertani y Weigle, 1953). Esto ocurre debido a la presencia de sistemas que reconocen DNA modificado, que actúan como sistemas "inmunes" muy primitivos. Solamente han sido descritos sistemas de este tipo en organismos unicelulares (principalmente bacterias) y sus viruses (Roberts y Macelis, 1994). Su función principal aparentemente consiste en proteger al organismo de DNA extraño: si éste se encuentra, es degradado (Noyer-Weidner y Trautner, 1993). Para evitar que el DNA propio se degrade, necesita haber una manera de distinguirlo del ajeno. De hecho existen dos maneras de lograr esta distinción, y ambas ocurren. Una manera consiste en identificar bases modificadas y degradar el DNA que los contiene. Los sistemas McrA, McrBC ("methyl citosine restricting") y Mrr ("modified adenine recognition and restriction") cumplen precisamente esta función en *E. coli* (Noyer-Weidner y Trautner, 1993; Redaschi y Bickle, 1996). McrA y McrBC anteriormente habían sido descritos como RglA y RglB ("restricts glucoseless DNA") ya que degradan el DNA de fagos mutantes que contienen 5-hidroximetilcitosina (hm5C) en lugar de la versión glucosilada que presentan los fagos silvestres (Revel y Luria, 1970). Ahora se sabe que además de cortar DNA con hm5C, también cortan ciertas secuencias que contienen m4C y m5C. Mrr corta secuencias de DNA con m6A, aunque puede actuar sobre m5C (Waite-Rees, et al. 1991). La especificidad de secuencia de este sistema tampoco está claramente descrita. La otra manera de distinguir entre el DNA propio y el ajeno es modificar el primero y tener una actividad que degrade todo lo que no se encuentra modificado. Los sistemas que cumplen esta función son los llamados de restricción y modificación o R-M (Noyer-Weidner y Trautner, 1993; Redaschi y Bickle, 1996). La modificación es llevada a cabo por enzimas (metilasas o metiltransferasas) que reconocen secuencias definidas de DNA y, utilizando S-adenosil metionina (SAM) como donador, transfieren un grupo metilo a una citosina o adenina específica dentro de esta secuencia. La actividad de restricción ocurre cuando la contraparte del sistema reconoce el mismo sitio y, si éste no se encuentra metilado, corta el DNA (estas enzimas se conocen como endonucleasas de restricción). Estos sistemas se pueden catalogar de acuerdo a sus

subunidades, los cofactores que requieren y la posición en que cortan el DNA en tres tipos: I, II y III.

Los sistemas R–M de tipo I están formados por tres subunidades, cada una especializada para cumplir una tarea dentro de un mismo complejo enzimático. Una subunidad se encarga de metilar adeninas (M), otra de cortar (R) y una tercera se encarga del reconocimiento del sitio específico (S) (Yuan y Hamilton, 1984). Los sitios reconocidos por estas enzimas son asimétricos, formados por 3 pb específicas seguidos por un espaciador variable de 6–8 pb y luego otras 4–5 pb específicas. El sitio de corte puede encontrarse lejos del sitio de reconocimiento, raramente a menos de 400 pb, pero inclusive hasta 7,000 pb de distancia. La enzima requiere SAM para unirse al DNA, y por lo tanto éste funciona como activador alostérico además de ser el donador de metilos. La endonucleasa también requiere ATP y Mg^{2+} para actuar. La naturaleza modular de estos sistemas les permite evolucionar vía recombinación y probar nuevas especificidades modificando solamente la subunidad S, que se encarga de la especificidad de ambas reacciones, metilación y restricción. En el genoma de *E. coli* K–12, los genes que codifican para este sistema (*hsdR*, *M* y *S*; "host specificity for DNA") se encuentran agrupados con *mrr* y *mcrBC*, por lo cual se le ha nombrado "región de control inmigratorio" a esta zona (Redaschi y Bickle, 1996).

Los sistemas R–M de tipo II constan de dos enzimas independientes, una encargada de la metilación y otra de la restricción (Noyer–Weidner y Trautner, 1993). Estas últimas son las famosas enzimas de restricción utilizadas en aplicaciones de biología molecular. Estas enzimas dependen de Mg^{2+} y sus correspondientes metilasas de SAM. Las bases metiladas que producen estos sistemas son m4C, m5C y m6A. La mayoría de los sitios que reconocen son simétricos (palíndromes) de 4–8 pb y en algunos casos se encuentran interrumpidos por espaciadores. Tanto el corte como la metilación ocurren simétricamente sobre el sitio de reconocimiento. Sin embargo, mientras que la endonucleasa generalmente requiere ser un homodímero para cortar conjuntamente ambas cadenas del DNA, la metilación ocurre en una cadena a la vez, actuando la metilasa como monómero. Esto sugiere una manera en que se facilitaría la entrada de este tipo de sistema a un nuevo organismo. La metilasa puede empezar a actuar inmediatamente sobre el DNA, protegiéndolo, mientras que la restricción requiere de una mayor concentración de enzima para lograr formar dímeros, y por lo tanto de más tiempo (Redaschi y Bickle, 1996). Existe además una subclase llamada IIS ("shifted cleavage") cuyo sitio de reconocimiento es asimétrico y el corte ocurre afuera de éste, pero

que se parece en los otros aspectos a los sistemas de tipo II (Szybalski, et al. 1991).

Los sistemas R–M de tipo III se asemejan a los del tipo I en que forman una sola enzima multifuncional (Redaschi y Bickle, 1996). Sin embargo, tienen sólo dos subunidades, Mod y Res que se encargan de la modificación y restricción respectivamente. La metilación requiere de SAM y la restricción requiere Mg^{2+} y ATP, pero estos factores además estimulan su contraparte, causando que en presencia de los tres y DNA no modificado, ambas reacciones compitan (Haberman, 1974). Los sitios que reconocen son asimétricos de 5–6 pb y el corte ocurre a unos 25 nucleótidos hacia su lado 3'. Res requiere de Mod para funcionar ya que Mod es el que le confiere la especificidad por el sitio de DNA. Mod puede actuar solo, al igual que un complejo de M y S en los sistemas de tipo I. Aunque pareciera que este tipo de sistema solamente metila la adenina de una cadena, aparentemente se requieren dos secuencias de reconocimiento, una en cada cadena y en sentido inverso. En conjunto, estos dos actúan como un sitio simétrico interrumpido, evitando así que surja DNA sin metilar (sustrato de la restricción) al pasar la horquilla de replicación (Meisel, et al. 1992).

Hay algunos sistemas, especialmente dentro de los ahora designados como del tipo IIS, que presentan muchas irregularidades por lo que posiblemente se vaya a requerir modificar el presente sistema de clasificación (Janulaitis, et al. 1992).

Metilación en *E. coli*

El DNA en *E. coli* K–12 contiene dos bases metiladas: m5C y m6A. Aproximadamente el 1.5% de las adeninas y el 0.75% de las citosinas se encuentran modificadas de esta manera (Marinus, 1996). Además de la metilación de adeninas por HsdM (el sistema R–M de tipo I que metila la segunda adenina de la secuencia AAC(6N)GTGC, representando el 0.02% de las adeninas totales), existen dos metilasas más: Dam ("DNA adenine methylase") que metila las adeninas en GATC y Dcm ("DNA cytosine methylase") que metila la segunda citosina de CC[A/T]GG (Marinus, 1987). Estas secuencias metiladas se encuentran en la mayoría de las enterobacterias, pero cabe mencionar que Dcm no es tan común como Dam ya que inclusive en *E. coli* B, ésta no se encuentra presente. El hecho de que se hayan aislado triples mutantes *hsd dam dcm* implica que estas funciones no son indispensables para la viabilidad de la bacteria. No se ha detectado ningún tipo de metilación en estas mutantes, lo cual indica que son las únicas enzimas encargadas de esta función (Marinus, 1996). Lo curioso de Dam y Dcm es que, siendo enzimas de metilación, no existen enzimas

de restricción correspondientes (para completar el sistema R–M de tipo II). Por lo tanto, no parece que su importancia actual radique en la protección del cromosoma bacteriano. De hecho, *Eco*RII corta en el sitio de *dcm*, pero los plásmidos que lo producen tienen su propia metilasa (70% de secuencia de aminoácidos idéntica a Dcm) la cual se expresa antes que la endonucleasa así obviando una posible necesidad de Dcm (Marinus, 1996). Si la utilidad de estas dos enzimas no consiste en proteger al genoma de actividades de restricción, ¿cuál es entonces la presión evolutiva que las ha mantenido?

Funciones de Dam

Las mutantes en *dam* que se han aislado, aunque viables, presentan una serie de fenotipos entre las que destacan: elevación en la tasa de mutagénesis (Marinus y Morris, 1974), incremento de la transposición de algunos transposones (Lundblad y Kleckner, 1984), iniciación asíncrona de la replicación (Boye, et al. 1992) y alteración de la expresión de algunos genes (Barras y Marinus, 1989). Todos estos fenómenos, y otros, han sido explicados gracias a la dilucidación del papel que juega Dam en la fisiología de la bacteria.

Dam actúa justo detrás de la horquilla de replicación, metilando las cadenas recién sintetizadas. Sin embargo, esto no ocurre inmediatamente; hay una ventana temporal en que el DNA se encuentra hemimetilado, es decir, la cadena original se encuentra metilada, pero la nueva no. Así, cuando el aparato de replicación deja errores, existe una manera de distinguir precisamente la cadena que los contiene, de no ser así se repararían indistintamente ambas cadenas, dejando el error en la mitad de los casos (Wagner y Messelson, 1976). Estos apareamientos erróneos de bases son reconocidos y eliminados por un mecanismo que involucra a MutS, MutL y MutH, y que depende de la metilación por Dam (Modrich, 1991). MutS reconoce el sitio en que existe el error, que puede ser una base equivocada, o una inserción o deleción de hasta 4 nucleótidos (Parker y Marinus, 1992). Luego se une MutL y sirve para activar de alguna manera a MutH que es capaz de hacer cortes de cadena sencilla al lado 5' de una secuencia GATC sin metilar (Welsh, et al. 1987). De este modo MutH reconoce el sitio de Dam más cercano y, si éste se encuentra hemimetilado, corta la cadena no modificada. El DNA entre el sitio de corte y el apareamiento erróneo es degradado y resintetizado. Este mecanismo depende de un par de cosas que vale la pena mencionar. En primer lugar, Dam tiene que estar cuidadosamente regulado. Una sobreexpresión ocasiona que se metilen muy rápido los sitios, antes de que pueda ocurrir la reparación. Con un nivel

muy bajo de Dam, puede iniciarse un nuevo ciclo de replicación antes de que se hayan terminado de metilar todos los GATC, dando lugar a sitios sin metilar. En ambos casos desaparece el modo de distinguir la cadena recién sintetizada (Marinus, et al. 1984). El segundo punto, es la necesidad de que existan sitios GATC en todo el genoma para que pueda ocurrir este tipo de reparación. De hecho, se ha visto que a distancias mayores de dos kilobases entre dos secuencias GATC, el mecanismo ya no es eficiente (Modrich, 1991). Por lo tanto, si en una región determinada de DNA existen muchos sitios GATC, esta región podrá ser reparada eficazmente, mientras que entre menos haya, es más probable que un error pueda no ser corregido.

Una parte del cromosoma que contiene muchos sitios GATC (más de diez veces mayor al esperado) es el origen de replicación (Oka, et al. 1980). Su importancia aquí no radica en asegurar una buena reparación de la zona, mas bien tiene que ver con la sincronización del inicio de replicación en un proceso que depende de DnaA. También se asume que favorece la apertura de las hebras dada la menor estabilidad de apareamiento de la m6A (Yamaki, et al. 1988). En el primer paso del inicio de replicación se requiere que DnaA se pegue a varios sitios en el origen de replicación, *oriC* (Bramhill y Kornberg, 1988). Una vez que las horquillas se alejan, la región cercana a *oriC*, incluyendo el gen *dnaA*, quedan hemimetiladas, y en esta condición son atrapadas por un componente proteico de la membrana externa (Ogden, et al. 1988). Durante este estado de secuestro (30 al 40% del ciclo celular), el orígen y *dnaA* permanecen inactivos y por lo tanto no pueden iniciarse nuevos ciclos de replicación. Dam tiene que competir por el acceso al origen, y cuando logra metilarlo completamente, éste se desprende de la membrana y puede volver a activarse en cuanto haya suficiente DnaA (Campbell y Kleckner, 1990). Originalmente se creyó que la interacción con la membrana podía facilitar la segregación de los nuevos cromosomas, al ir creciendo la membrana podría ir arrastrando y separando ambos orígenes (Ogden, et al. 1988). Sin embargo, el hecho de que mutantes *dam* segreguen normalmente sus cromosomas, invalida esta suposición (Vinella, et al. 1992).

El hecho de que ciertas proteínas membranales tengan afinidad por el origen solamente en estado hemimetilado, nos habla de que existen componentes celulares que pueden interactuar con sitios GATC, específicamente dependiendo de su estado de metilación. Algunos de estos componentes podrían ser factores transcripcionales y por lo tanto la expresión de genes cuyos promotores contienen la secuencia GATC puede verse

acoplada al ciclo de replicación o afectada por el nivel de Dam. Algunos casos descritos de procesos dependientes de Dam son, la transposición de Tn*10* (Roberts, et al. 1985), la expresión del operón *pap* (Van der Woude, et al. 1993) y el empaquetamiento del fago P1 (Sternberg y Coulby, 1990).

Funciones de Dcm

Después de ver todos los procesos en los que se ve involucrado Dam, uno pensaría que Dcm podría participar de igual manera en la fisiología de *E. coli*. Sin embargo, no se ha detectado ningún fenotipo relacionado a la sub o sobreexpresión de Dcm (Marinus, 1996). No obstante, hay una propiedad curiosa de las citosinas metiladas que es necesario tomar en cuenta.

Las citosinas pueden sufrir desaminación espontáneamente, dejando la base uracilo en su lugar. La reparación de este daño depende de la enzima uracil–N–glucosilasa que reconoce uracilo en DNA y lo quita (Lindahl, 1982). El problema es que cuando una citosina metilada (producido por Dcm) se desamina, queda timina, una base normal del DNA. Para contrarrestar el efecto aparentemente mutagénico de *dcm*, existe un gen a su inmediato 3' (de hecho sobrelapado por 6 codones) llamado *vsr* ("Very Short Patch repair" o VSP) cuyo producto reconoce y remueve una timina cuando ésta se encuentra en el contexto del sitio de reconocimiento de Dcm y está mal apareada con guanina (indicando que era originalmente una citosina) (Hennecke, et al. 1991). El único problema con este sistema es que si la secuencia original realmente contenía T–A y una mutación azarosa cambió la adenina por guanina, el sistema VSP no va a corregir la situación. De hecho, el sistema se encargará precisamente de quitar la T (base correcta) y poner en su lugar una C (base incorrecta) logrando así la fijación de la mutación. Además, el sistema va a estar en algunos casos en competencia con la reparación dirigida por GATC que repararía el error ciegamente, de acuerdo a la cadena recién sintetizada (Welbauer, et al. 1993). El resultado final es que en secuencias que contienen CC[A/T]GG, la segunda citosina es altamente susceptible de ser mutagenizada ("hotspot" mutagénico, Duncan y Miller, 1980).

ANTECEDENTES

Varios trabajos se han enfocado a estudiar la distribución de los sitios de metilación Dam y Dcm en el genoma de *E. coli* (Barras y Marinus, 1988; Bhagwat y McClelland, 1992; Merkl, et al. 1992; Gómez-Eichelmann y Ramírez-Santos, 1993). La importancia de este tipo de análisis radica en la dilucidación de las funciones biológicas que desempeñan estas secuencias modificadas. Por ejemplo, la hipótesis de que los sitios GATC tienen que ver con la regulación del inicio de replicación en *E. coli*, viene en parte de la observación de que existen 11 de estos sitios en el origen mínimo *oriC* (Zyskind y Smith, 1986). Esto representa una frecuencia más de diez veces mayor a la esperada dado el tamaño de esta secuencia (232 pb).

Barras y Marinus buscaron regiones con una alta frecuencia de sitios GATC y regiones en las que esta secuencia no estuviera presente (Barras y Marinus, 1988). A este tipo de distribución le llamaron de "clusters" (agrupamientos) y "voids" (vacíos) y los definieron arbitrariamente de la siguiente forma: un vacío es una región de al menos 600 nucleótidos que no presenta la secuencia GATC y un agrupamiento ocurre cuando al menos 3 GATC se encuentran separados por menos de 30 nucleótidos o al menos 4 GATC separados por menos de 45 nucleótidos. Pudieron observar que *E. coli* presenta diversos agrupamientos y vacíos a lo largo de su genoma. El agrupamiento mayor que encontraron fue de 9 GATC y presentó una frecuencia de 1 sitio cada 25 nucleótidos. El vacío más grande fue de 1618 nucleótidos, por lo que concluyeron que el genoma de *E. coli* se encuentra totalmente protegido por el sistema de reparación dependiente de Dam (ver Introducción; Modrich, 1991). La frecuencia total de los sitios Dam en las regiones que analizaron resultó ser de 1 cada 222 nucleótidos. Sin embargo, al tomar en cuenta por separado las regiones transcritas y traducidas (*codificante*), transcritas pero no traducidas (*sólo transcritas*: tRNA, rRNA, etc) y las que no son transcritas ni traducidas (*intergénicas*), encontraron que las regiones con la mayor densidad de sitios GATC eran las traducidas y que aquellas regiones transcritas pero no traducidas (rRNA, tRNA) presentaban la menor densidad. Para explicar estos hallazgos propusieron que dada la interacción de la secuencia GATC con ciertas proteínas (como ocurre en el inicio de replicación), se vería desfavorecida su selección en la mayoría de las

secuencias reguladoras. Además, dado que GATC es un palíndrome, podría formar una estructura secundaria no deseada para aquellos RNA donde precisamente esta estructura tiene que estar finamente controlada. Sin embargo, las secuencias que utilizaron sumaban solamente 79,333 nucleótidos (1.7% del cromosoma) por lo que no podemos considerar sus conclusiones como definitivas para el genoma completo de *E. coli*.

Como se describe más a fondo en la Introducción de esta tesis, Dam está relacionado con un mecanismo de reparación y por tanto puede reducir la tasa de mutaciones espontáneas de secuencias que lo contienen. Por otro lado, la citosina metilada formada por Dcm es propensa a desaminarse para dar lugar a timina, aumentando la tasa de mutaciones espontáneas del DNA que la contiene. Gómez-Eichelmann y Ramírez-Santos analizaron la distribución de CCWGG en *E. coli* y trabajaron con la hipótesis que la frecuencia de ambos tipos de metilación, Dam y Dcm, podría definir dos grupos de genes con diferentes tasas de cambio (Gómez-Eichelmann y Ramírez-Santos, 1993). Aquellos genes con una alta frecuencia de CCWGG pero baja frecuencia de GATC podrían cambiar a una velocidad mayor que genes con frecuencias cercanas al promedio. Así mismo, genes con altas frecuencias de GATC pero bajas de CCWGG podrían cambiar con una tasa mutagénica menor. Para este trabajo utilizaron los 207,530 nucleótidos (4.5% del cromosoma) que sumaban los fragmentos continuos más grandes secuenciados hasta ese momento. De su análisis estadístico encontraron que la frecuencia de los sitios de metilación era de 1 sitio Dcm cada 351 nucleótidos y 1 sitio Dam cada 214 (muy cercano al resultado de Barras y Marinus, 1988). El vacío mayor de CCWGG fue de 1,869 nucleótidos y el agrupamiento mayor (definido en este caso como 200 o menos nucleótidos con al menos 3 sitios Dcm) fue de 3 CCWGG en 33 nucleótidos. Además, en un análisis de 55 genes, observaron que la citosina metilada caía más frecuentemente en la primera posición de codones (64%) que en segunda (17%) o tercera (19%) posición. Para atacar la hipótesis de las diferentes tasas de mutación, buscaron genes con aumentos o decrementos de dos veces la frecuencia de los sitios de metilación. Solamente un gene pasó su criterio (*uncF*), con una frecuencia baja de Dam y alta de Dcm. De las otras combinaciones de frecuencias no encontraron un sólo caso. Dados estos resultados, concluyeron que la tasa de mutagénesis de pocos genes podría ser modificada por la relación de las frecuencias de sus sitios de metilación.

Aunque las secuencias con las que trabajaron Gómez-Eichelmann y Ramírez-Santos cubrían más del doble que las de Barras y Marinus, hay que recalcar que sólo representan

una fracción muy pequeña (4.5%) del cromosoma de *E. coli*. Algunas de las conclusiones a las que llegaron estos y otros autores forzosamente deben ser consideradas como tentativas, hasta que los análisis puedan extenderse a la mayor parte del genoma. En la actualidad se cuenta con la secuencia completa del genoma de *E. coli* K–12 (Blattner, et al. 1997) lo que nos permite retomar estos problemas con una perspectiva mucho más amplia, además de poder abordar cuestiones que anteriormente, simplemente no eran posibles.

OBJETIVOS

- Calcular la frecuencia y distribución de los sitios Dam (GATC) y Dcm (CCWGG) en el genoma de *E. coli*. Comparar estos resultados con la frecuencia esperada al azar y por un análisis markoviano, buscando evidencia de una presión selectiva sobre este tipo de secuencias.

- Buscar patrones en la cantidad de sitios de ambas metilaciones en ventanas discretas alrededor del cromosoma de *E. coli*.

- Encontrar los "vacíos" más grandes de sitios Dam y Dcm, el primero para ver si realmente todo el genoma está protegido y el segundo para buscar regiones que no tengan una carga mutagénica importante por la citosina metilada.

- Buscar agrupaciones estadísticamente significativas de ambas secuencias de metilación para localizar regiones de posible importancia biológica.

- Obtener y comparar la frecuencia de los sitios de metilación en distintas regiones del genoma: secuencia transcrita y traducida (*codificante*), aquella que solamente se transcribe (*sólo transcrita*) y secuencia que no se transcribe (*intergénica*).

- Calcular la frecuencia promedio de aparición de las bases metilables en cada posición de todos los codones del genoma.

- Organizar todos los genes de *E. coli* de acuerdo a su contenido de sitios de metilación. Averiguar si genes de distintas categorías, como esenciales y no esenciales, presentan proporciones distintivas de estos sitios.

MATERIAL Y MÉTODOS

Materiales

En el presente trabajo se utilizó la secuencia completa de *Escherichia coli* K–12 en formato GenBank que se encuentra disponible en internet (Benson, et al. 2000). Para facilitar el acceso a la secuencia, este archivo fue procesado para dejar un archivo únicamente con la secuencia nucleotídica lineal (Programa 01, ver Programas). También se utilizó el genoma completo codificado en formato Sensa (Ciria y Merino, 2001). Sensa es un programa que toma la anotación de un genoma en formato GenBank y crea un archivo de una sola línea donde cada caracter representa una sola base, pero ahora incluye la información proveniente de la anotación, en efecto comprimiéndola y facilitando muchísimo su manejo. La información que contiene cada nueva base incluye el tipo de gen al que pertenece, la cadena en la que se transcribe ese gen, si tiene o no otro gen sobrelapado, en que posición de codón se encuentra y obviamente, el tipo de nucleótido. La lista depurada de los genes traducidos utilizada para algunos de los análisis fue proporcionada amablemente por el Dr. Gabriel Moreno del CIFN, UNAM. Las listas de genes esenciales y no esenciales para *E. coli* fueron tomadas de una base de datos japonesa, SHIGEN ("Shared Information of Genetic resources", http://www.shigen.nig.ac.jp/ecoli/pec/Analyses.jsp). Todos los programas utilizados fueron escritos específicamente para contestar las preguntas planteadas y se encuentran en la sección de Programas. Se utilizó Perl 5 (http://www.perl.com/pub) como lenguaje de programación, teniendo por computadora una PC con Mandrake Linux como sistema operativo (http://www.linux-mandrake.com/en/). Para los diagramas circulares se aprovechó una librería gráfica para Perl llamada GD (http://stein.cshl.org/WWW/software/GD/GD.html). Todo el texto, la organización de las gráficas, la creación de las tablas y otros detalles cosméticos fueron realizados en StarOffice (http://www.sun.com/products/staroffice/get.html).

Metodología general

Los objetivos planteados se cubrieron a tres niveles. En el primero se dividió al genoma completo de *E. coli* en ventanas discretas de distintos tamaños. Para el segundo

Metodología específica – *enfoque funcional*

Los últimos tres objetivos (ver Objetivos) se cubrieron dentro de esta metodología. Dos objetivos fueron de un carácter más general. Primero, se consideraron tres grandes regiones funcionales del genoma de *E. coli:* regiones codificantes, sólo transcritas e intergénicas. Con el Programa 13 se separaron estas regiones y se obtuvo la frecuencia correspondiente de cada sitio de metilación. El programa Sensa (Ciria y Merino, 2001) fue sumamente útil para extraer la información necesaria de las anotaciones del GenBank y dejarlo en un formato utilizado por el Programa 13. Para tomar en cuenta el posible efecto del porcentaje de G–C sobre estas frecuencias fue también necesario obtener el uso de nucleótidos para cada una de estas regiones (Programa 14). En segundo lugar se calculó la frecuencia de aparición de las bases metilables en cada posición de los codones, de nuevo usando un genoma codificado en formato Sensa. Aprovechando el genoma lineal de ceros y unos para las posiciones de los sitios de metilación (generado con el Programa 10), se buscó cada posición en la base de datos de Sensa para averiguar si se trataba de primera, segunda o tercera base de un codón (Programa 15). Cuando dos genes sobrelapaban y la base metilable representaba dos posiciones de dos codones diferentes, se contó como una ocurrencia independiente de cada caso. Además, dada la naturaleza palindrómica de los sitios de metilación, se procuró contar las dos bases metilables de cada secuencia. Se contó también la cantidad de bases metilables que corresponden a posiciones intergénicas así como las que corresponden a genes no traducidos. En este caso no fue necesario obtener el porcentaje de G–C para las tres posiciones de cada codón ya que Ricardo Ciria (autor de Sensa) ya los había obtenido y amablemente nos las proporcionó.

Para cumplir con el último objetivo, se utilizó la base de datos de genes depurados y se tomó cada gene por separado. En este caso, el tipo de análisis realizado para los dos tipos de metilación fue diferente. Para Dcm simplemente se contaron los sitios presentes en la región codificante, ya que el efecto de este tipo de metilación es solamente local (Programa 16). Sin embargo, el mecanismo de reparación asociada a Dam actúa lejos del sitio (ver Introducción) por lo que cada sitio repercute sobre un entorno de 1,000 bases en ambas direcciones (Modrich, 1991). Por ello se tomó cada una de las bases individuales de cada gene y se analizaron 1,000 bases hacia arriba y 1,000 bases hacia abajo, para ver cuantos sitios Dam podían tener efecto sobre esa base (Programa 16). Ambos resultados fueron reunidos para cada uno de los genes y posteriormente graficados. Este mismo análisis se

realizó para dos grupos de genes tomados de SHIGEN ("Shared Information of Genetic resources", http://www.shigen.nig.ac.jp/ecoli/pec/Analyses.jsp) (Programas 17). El primer grupo consistía en 201 genes esenciales y el segundo consistía en 297 genes cuya función no era esencial. Se realizaron las gráficas para compararlas entre sí y contra el de los genes totales.

RESULTADOS Y DISCUSIÓN

Frecuencia de los sitios de metilación

Como se comentó en los Antecedentes del presente trabajo, uno de los primeros datos que se suele obtener de las secuencias Dam y Dcm es su frecuencia. Estas frecuencias se expresan generalmente como la cantidad promedio de nucleótidos que presenta un sitio, con este tipo de modificación, en un intervalo de secuencia determinada. Los resultados de los diferentes trabajos han concordado en este respecto, a pesar de variar la cantidad de secuencia analizada (Barras y Marinus, 1988 con 1.7% del cromosoma y Gómez-Eichelmann y Ramírez-Santos, 1993 con 4.5% del cromosoma). El primer paso fue entonces averiguar si este panorama se mantiene con el genoma completo. Las frecuencias de estas secuencias las calculamos simplemente contando la cantidad de sitios, tomando como ventana la totalidad del genoma (Programa 02). El programa puede utilizarse para contar la cantidad de sitios de cualquier secuencia presente en un genoma, dividido en ventanas de un tamaño especificado. Para interpretar estos números, es necesario compararlos con los datos generados por un método estadístico, precisamente para averiguar si son significativos. El método más sencillo consiste en utilizar las frecuencias o probabilidades de los elementos unitarios de la secuencia a buscar. Esto es, la probabilidad de encontrar la secuencia GATC es simplemente el producto de las probabilidades de sus elementos G, A, T y C; expresado como:

$$p(GATC \mid G,A,T,C) = p(G) \cdot p(A) \cdot p(T) \cdot p(C)$$

Para cualquier secuencia compuesta por proporciones iguales de cada base (como es prácticamente el caso del genoma de *E. coli*), cualquier tetranucleótido tendría así una frecuencia esperada de $(¼)^4$, o 1 en 256. Sin embargo, este cálculo como método predictivo es pésimo, ya que existen en el genoma de *E. coli* secuencias como CTGG que están representadas 1 en 137 y otras como CTAG que tan sólo son encontradas cada 5230 pb. Este y otros métodos para predecir la frecuencia de oligonucleótidos en secuencias del genoma de *E. coli* ya han sido evaluados y la cadena de Markov resultó ser el mejor (Phillips, et al. 1987). Una cadena de Markov toma en cuenta la frecuencia de los oligos que componen

la secuencia de búsqueda, esto es, para GATC se toma en cuenta la frecuencia observada de GAT, de ATC y de la secuencia que los une, AT. Expresado matemáticamente:

$$p(GATC|GAT, ATC) = \frac{p(GAT) \cdot p(ATC)}{p(AT)}$$

y para un pentanucléotido, en este caso CCWGG:

$$p(CCWGG|CCWG, CWGG) = \frac{p(CCWG) \cdot p(CWGG)}{p(CWG)}$$

donde p(X) representa la frecuencia observada de la secuencia X, y p(X|Y) la frecuencia esperada de X corrigiendo para la frecuencia observada de Y.

En la Tabla 1 se muestran las frecuencias observadas de ambas secuencias de metilación y los datos esperados generados por el método de la cadena de Markov así como la tasa o relación entre los observados y esperados.

***Tabla 1. Frecuencias de sitios de metilación en* E. coli**

Secuencia	*Datos observados*	*Datos esperados*	*Tasa Obs/Esp*
GATC	1 en 243	1 en 192	0.79
CC[A/T]GG	1 en 385	1 en 279	0.73

Este tipo de comparación nos permite concluir que las secuencias GATC y CCWGG se encuentran subrepresentadas en el genoma de *E. coli*. Además, como la manera de calcular los datos esperados incluye la frecuencia observada de los componentes, el posible fenómeno biológico causa de esta reducción de frecuencia tiene que estar actuando a nivel de la secuencia completa. Esto nos sirve para descartar el efecto del uso de codones ya que actúa a nivel de trinucleótidos. Algo que debemos notar de ambas secuencias es que son palíndromes, esto es se "leen" igual en ambos sentidos. Esto podría representar una explicación de la reducción observada de frecuencias, ya que dos secuencias palindrómicas cercanas son capaces de formar estructuras secundarias de forma de asa, por lo que su presencia inapropiadamente regulada en secuencias que se transcriben a RNA de cadena sencilla (90% del genoma de *E. coli*) puede resultar perjudicial. Para probar esta hipótesis

calculamos las tasas de frecuencia observada/esperada para los 256 tetranucleótidos de igual manera que se hizo para GATC. Luego, las separamos en dos grupos, aquellos que son palíndromes y aquellos que no y se les calculó la media y la desviación estándar para las tasas de cada grupo. Los resultados pueden verse en la Tabla 2.

Tabla 2. Tasas de frecuencia esperada / observada de tetranucleótidos		
Tipo de tetranucleótido	***Tasa media***	***Desviación estándar***
No palíndrome	0.99	0.15
Palíndrome	0.84	0.20

Como es de esperarse, el grupo que incluye a prácticamente todos los tetranucleótidos es muy bien predicho por la cadena de Markov (tasa de 0.99, muy cercana a 1), sin embargo el grupo de los palíndromes se encuentra subrepresentado en el genoma, lo que implica que existe una presión de selección negativa para este tipo de secuencia.

La secuencia CCWGG se encuentra aún menos representada que GATC, y esto podría parecer algo contradictorio ya que el mecanismo de reparación por *vsr* (VSP repair, ver Introducción) se encuentra constantemente generando los tetranucleótidos CCWG y CWGG que deberían aumentar la frecuencia de CCWGG. El hecho de que esto realmente ocurra lo podemos observar comparando la frecuencia observada de CCWGG con la esperada por sus componentes unitarios y al calcularlo, la tasa es de 1.33. Sin embargo la cadena de Markov toma en cuenta los componentes, en este caso CCWG y CWGG, así que la tasa menor a uno de la Tabla 1 implica que existe algo más que está reduciendo la frecuencia específicamente actuando sobre CCWGG. Un mecanismo que explica en parte este fenómeno es precisamente la metilación por Dcm. En vista que la citosina metilada se desamina dando lugar a timina, la secuencia CCWGG irá desapareciendo para formar CTWGG si no es reparada. La metilación por Dam no afecta a la secuencia GATC ya que la modificación de la adenina no la hace más inestable.

En general, los datos observados con la secuencia completa del genoma se encuentran muy cercanos a los reportados anteriormente: un sitio Dam en 214 y un sitio Dcm cada 351 nucleótidos (Gómez-Eichelmann y Ramírez-Santos, 1993). Pero como se muestra en la Tabla 1, ambas secuencias se encuentran un poco menos representadas de lo que se creía.

Análisis por ventanas

Para empezar a estudiar los sitios de metilación en el genoma completo de *E. coli* decidimos analizar la secuencia en fracciones iguales, o ventanas. La ventaja de trabajar así es que no se imparten sesgos más allá de la selección del tamaño de ventana. Para disminuir lo más posible este sesgo escogimos 5 tamaños de ventana, 2.5, 5, 10, 20 y 40 mil pb explorando así un amplio rango de tamaños, desde uno que podría contener un sólo gen, hasta el de 40 mil pb, que es prácticamente un minuto del cromosoma. Con estas ventanas, evitamos el tener que fijar *a priori* un criterio de selección, como la búsqueda de una densidad determinada de sitios o la localización con respecto a genes o con respecto al inicio de replicación. Además, el escoger ventanas de un mismo tamaño nos permite visualizar fácilmente su distribución a lo largo del cromosoma y buscar regiones (cuyo tamaño o densidad no necesitamos saber de antemano) con datos sobresalientes.

Cuando se desea trabajar estadísticamente con una serie de datos es conveniente saber a que tipo de distribución se ajustan. Se sabe que muchos datos biológicos, especialmente cuando se tiene una gran cantidad de ellos, se ajustan bastante bien a una *distribución normal*, por lo que éste es el primer tipo de distribución que debemos tomar en cuenta. La fórmula matemática que representa a este tipo de distribución es la siguiente:

$$f_i = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(X_i-\mu)^2}{2\sigma^2}}$$

Donde f_i es la frecuencia de aparición del dato X_i, cuando los datos se distribuyen con una media igual a μ y una desviación estándar igual a σ. En la Figura 1 se muestra la gráfica de una distribución normal además del acercamiento a su cola positiva, donde se ve el área que representan aquellos datos mayores a tres desviaciones estándar. Estos datos sólo representan el 0.135% de la población y dado que la distribución es simétrica, lo mismo ocurre con los datos menores a tres desviaciones estándar.

Para ver si nuestros datos se ajustaban a este tipo de distribución, elaboramos histogramas para cada tipo de metilación y para cada tamaño de ventana. En cada gráfica también trazamos la curva teórica de una distribución normal, por lo que fue necesario calcular la media y desviación estándar para cada caso. Normalizamos las curvas teóricas

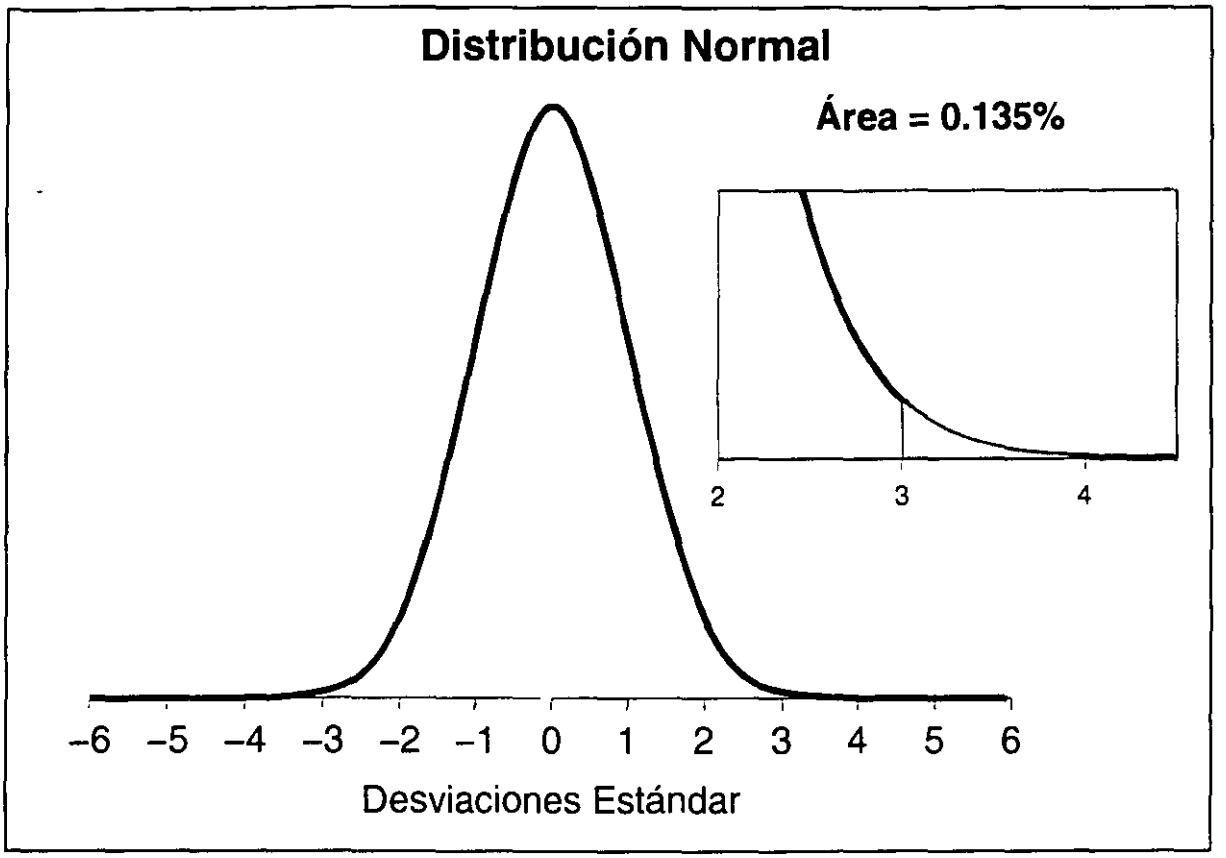
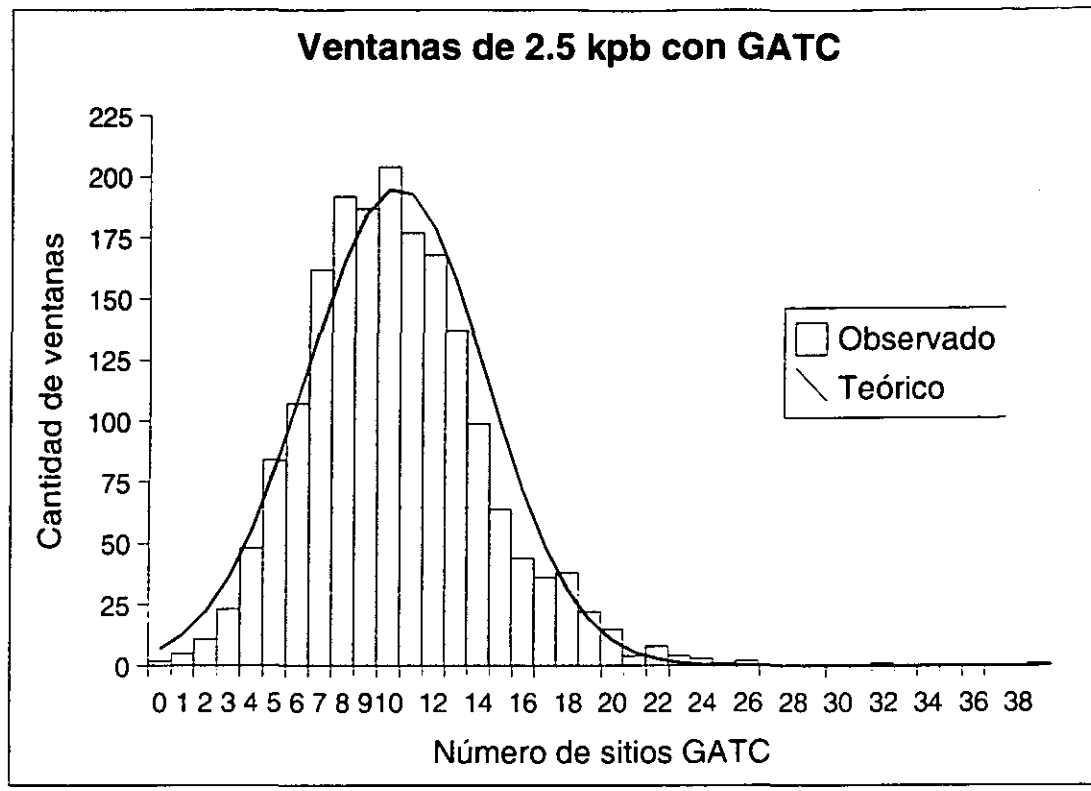


Figura 1. Porción de una distribución normal por arriba de 3 σ

multiplicando sus valores de frecuencia por una constante de ajuste. Este valor se obtuvo para cada caso y simplemente ajusta la frecuencia teórica de la media para que tome el mismo valor que la frecuencia observada de la media. En algunos casos fue necesario agrupar dos o tres valores observados en cada columna del histograma. Esto fue necesario ya que al tener diversos tamaños de ventana, por ende se tiene para cada caso una cantidad de datos diferente. Para que los histogramas fueran más comparables entre si, y con la distribución teórica, convenía que la cantidad de columnas de cada histograma fuera lo más semejante posible. En aquellos casos donde resultaban más del doble de columnas que las primeras gráficas, se agruparon de dos en dos; para más del triple de columnas se agruparon de tres en tres. A continuación se muestran los histogramas para los cinco tamaños de ventana y en cada caso para ambos tipos de secuencia de metilación (Figuras 2–6):



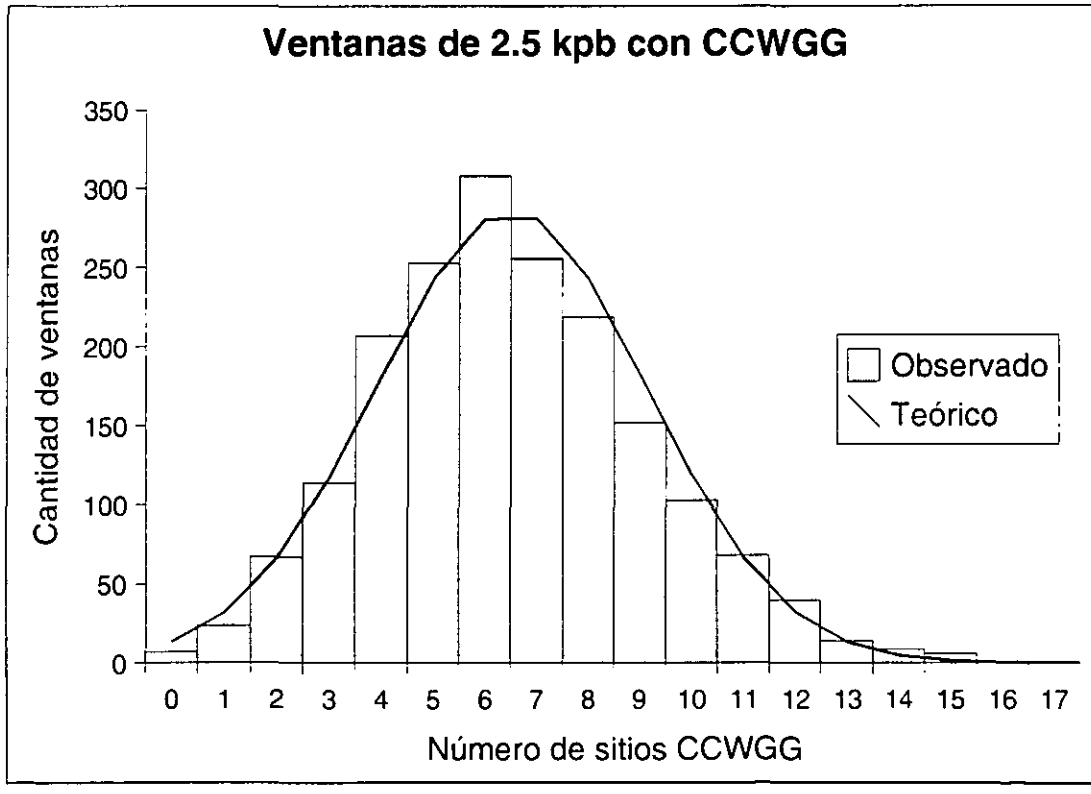


Figura 2. Histograma de frecuencias acumuladas utilizando ventanas de 2,500 pb

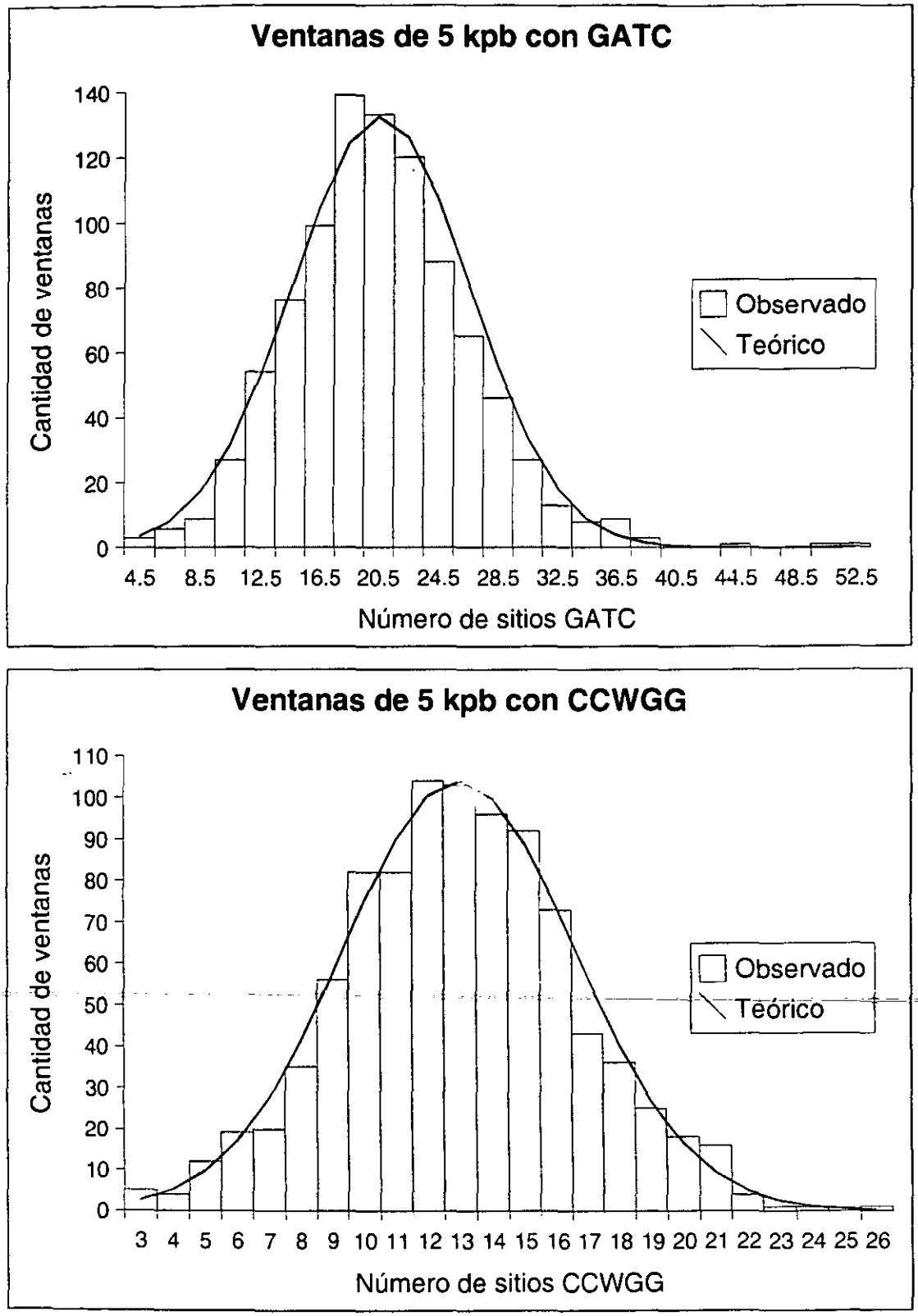


Figura 3. Histograma de frecuencias acumuladas utilizando ventanas de 5,000 pb

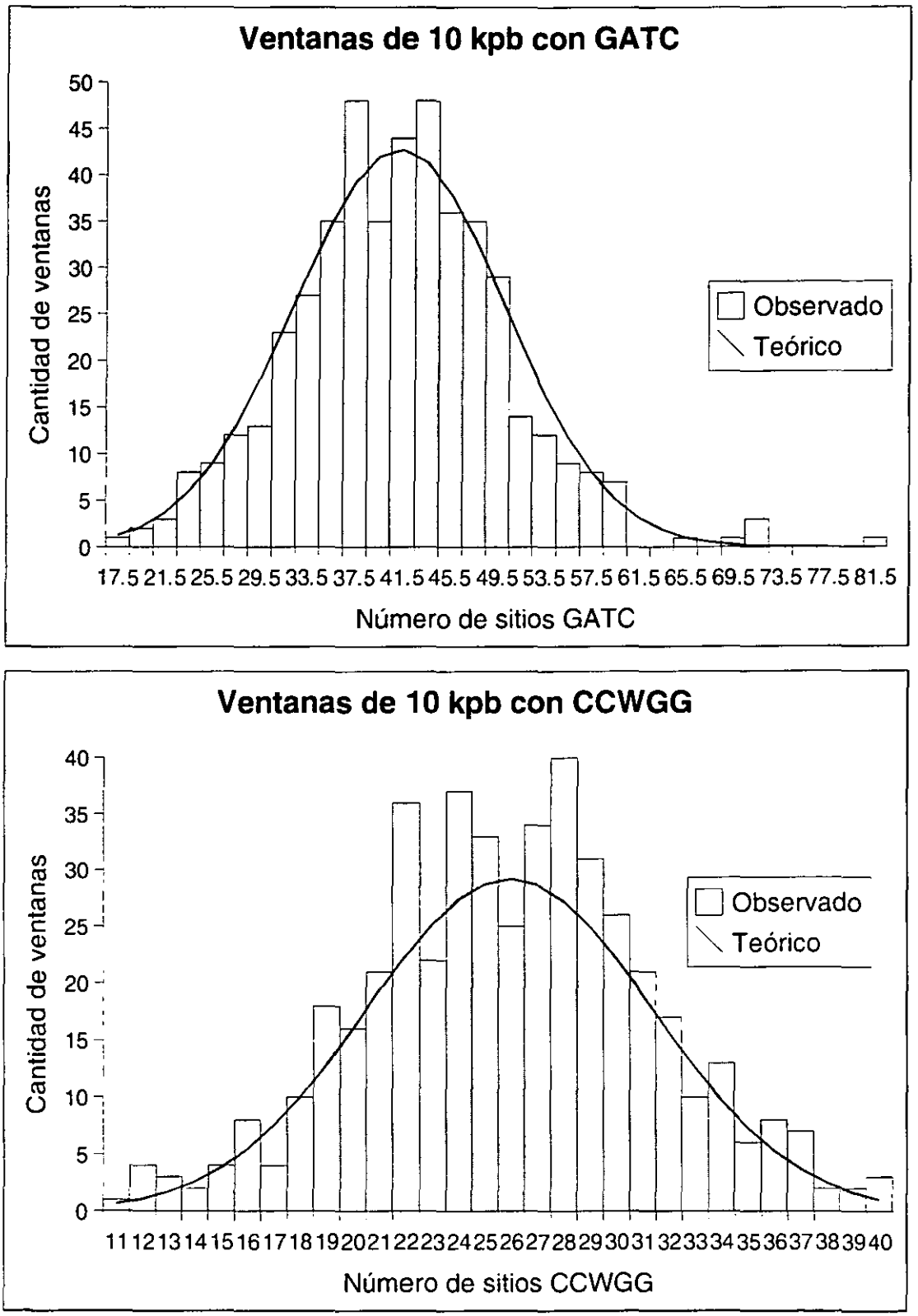


Figura 4. Histograma de frecuencias acumuladas utilizando ventanas de 10,000 pb

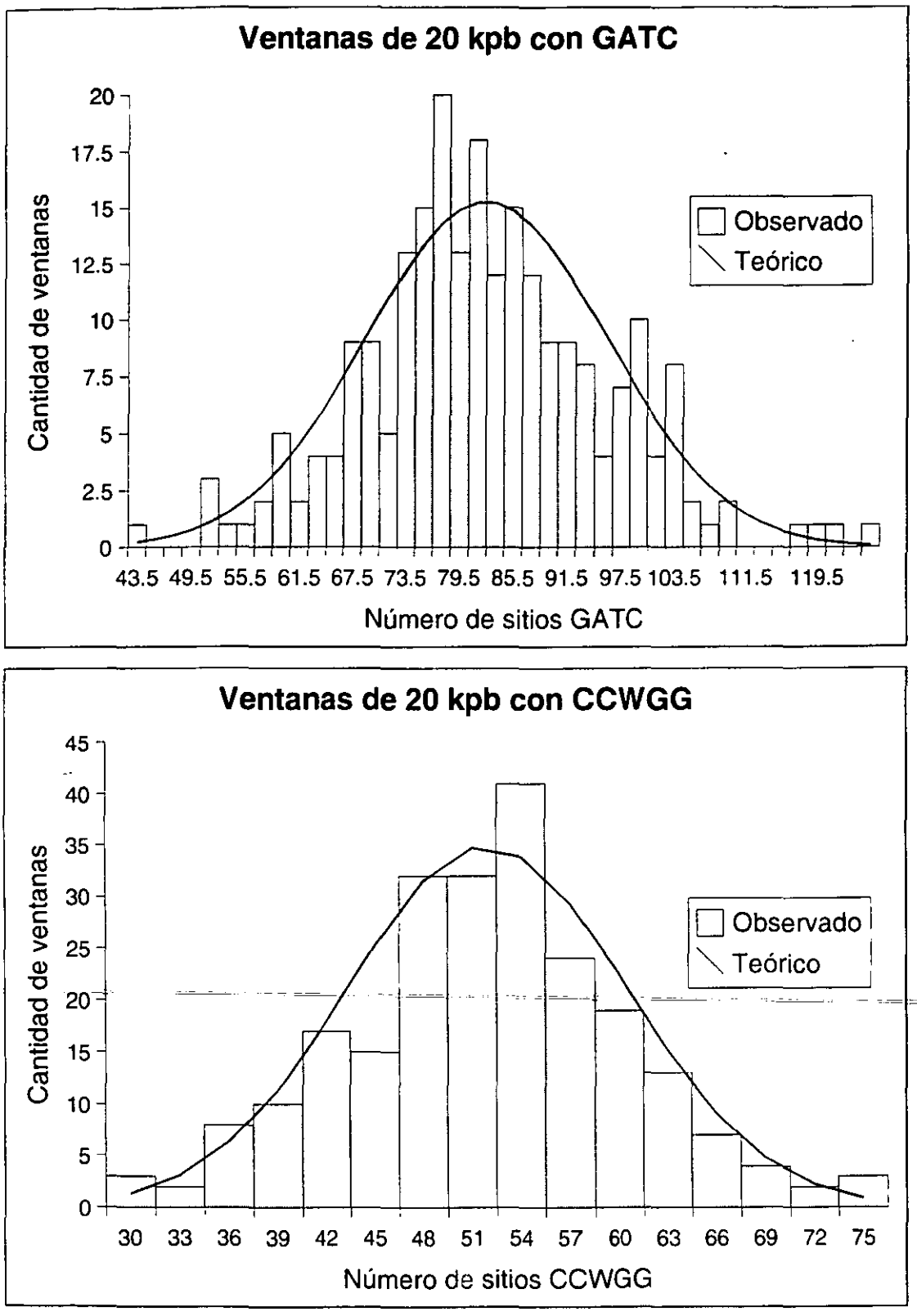


Figura 5. Histograma de frecuencias acumuladas utilizando ventanas de 20,000 pb

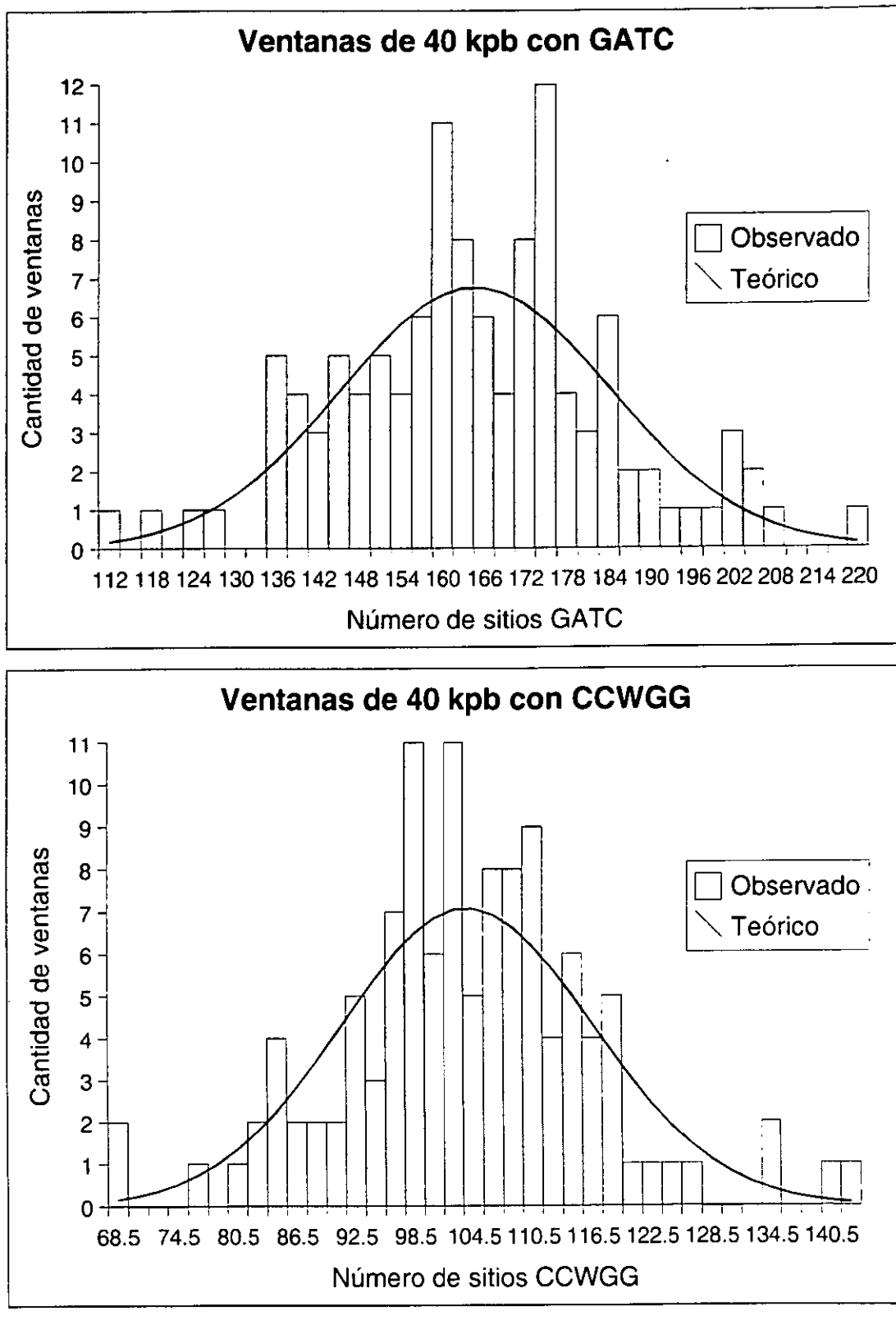


Figura 6. Histograma de frecuencias acumuladas utilizando ventanas de 40,000 pb

Como podemos ver, especialmente en las gráficas con mayor número de ventanas (Figuras 2 y 3), los datos experimentales se ajustan muy bien a una distribución normal. Sin embargo, vale la pena mencionar algunos detalles. En las primeras tres figuras de la distribución de las ventanas de GATC existen ventanas con valores que claramente se salen de la distribución. En la Figura 2 se puede ver claramente que las ventanas con 32 y 39 sitios GATC, caen más allá inclusive del rango de la media más tres desviaciones estándar ($\mu + 3\sigma$), que se suele escoger para encontrar datos significativos. Como recordamos del recuadro de la Figura 1, este rango equivale tan sólo al 0.135% de la distribución. De la misma manera, existen al menos tres ventanas de 5 mil pb y otras tres de 10 mil que podrían ser interesantes (Figuras 3 y 4). Esto no lo podemos observar en la cola izquierda de la distribución de sitios GATC, ni en alguna de las dos colas de la distribución de CCWGG. El caso de las colas izquierdas se debe a dos razones. Primero, cuando el tamaño de ventana es muy pequeño (2,500 pb), existen ventanas con cero sitios. Estas ventanas caen perfectamente dentro del rango $\mu - 3\sigma$ y dado que una ventana no puede presentar una cantidad negativa de sitios, es imposible que existan valores "significativos" en la cola izquierda. Aún en las ventanas de 5 mil pb la distribución es tal que solamente datos negativos se saldrían del rango mencionado. La segunda razón, evidente en las ventanas de 10 mil pb, es debido a que para ser significativas las ventanas tendrían que tener valores muy cercanos a cero. Esto representaría un vacío cercano a 10 mil pb, y esto, como veremos más adelante, no se observó. En las figuras de los dos tamaños mayores de ventana (Figuras 5 y 6), los datos experimentales ya no se ajustan tan bien a la distribución teórica. Esto se debe a que al aumentar el tamaño de ventana, disminuye la cantidad de datos y en estos dos casos los datos simplemente son insuficientes para ajustarse a una distribución teórica.

Los histogramas nos permiten concluir que efectivamente nuestros datos se distribuyen de una manera normal, siempre y cuando se cuente con una cantidad suficiente de ellos. Además nos hacen ver que no nos conviene tanto utilizar un criterio usual de selección de datos estadísticamente significativos (aquellos que caen fuera de $\mu \pm 3\sigma$) ya que tan sólo encontraríamos algunas ventanas de tamaño pequeño para el caso de GATC, y para CCWGG ni siquiera eso. Siguiendo con la idea propuesta inicialmente en la Metodología, de tratar de imponer la menor cantidad de sesgos posibles, decidimos considerar todas las ventanas y representarlas de una manera visual. Para ello creamos lo que llamamos un

diagrama genómico, que es un histograma circular cerrado representando el genoma de *E. coli*. El origen y término de replicación se encuentran indicados, así como el origen de secuenciación. Las flechas circulares externas representan la dirección de las horquillas de replicación y el círculo de mayor diámetro, es una representación de todos los genes, aquellos transcritos a favor de la horquilla son las barras externas, aquellos en contra de la horquilla, las barras internas. La escala utilizada se encuentra indicada en cada gráfica. Cada ventana es representada por una barra de histograma surgiendo de un círculo central. Aquellas barras que salen del círculo central hacia fuera representan ventanas con valores mayores al promedio, las barras que salen hacia el centro son menores al promedio. El valor numérico graficado, de acuerdo a la escala, fue calculado mediante una integración definida de la curva teórica de la distribución normal (utilizando los parámetros observados– media y desviación estándar) entre el valor de la ventana y el final de la cola de distribución. De esta manera, los valores representan la probabilidad de encontrar una ventana con al menos (para casos por arriba de la media), o a lo mucho (para casos por debajo de la media), la cantidad de sitios observados. Por lo tanto, la probabilidad de encontrar una ventana con al menos una cantidad de sitios igual a la media es de 0.5 o 50%. Para que se pudiera visualizar de mejor manera la mayor cantidad de ventanas, permitimos que valores muy altos (representando probabilidades muy pequeñas) se salieran de la escala, marcándolas con línea punteada e indicando su valor real. A continuación se presentan los diagramas genómicos de GATC y CCWGG para cada tamaño de ventana (Figuras 7–16). Además, se incluye una gráfica para cada tamaño de ventana donde el histograma representa los cuatro casos posibles de la combinación de los tipos de metilación, ambos tipos por arriba de la media, ambos por abajo, y los dos casos donde uno se encuentra por arriba y el otro por debajo de la media (Figuras 17–21). En estas últimas gráficas, la idea fue escoger las combinaciones de sitios de metilación tal que las barras que se encuentran hacia afuera del círculo podrían representar regiones estables, protegidas o poco mutagénicas; mientras que las barras "negativas" representan regiones con una posible tasa instrínseca de cambio mayor al promedio.

Como era de esperarse, las características que observamos en los histogramas se ven reflejados en los diagramas genómicos. Los diagramas para tamaños pequeños de ventana con GATC presentan unas pocas ventanas que claramente se salen de la distribución. Pero ahora podemos ver en que posición del genoma se encuentran localizadas este y cualquier otro detalle.

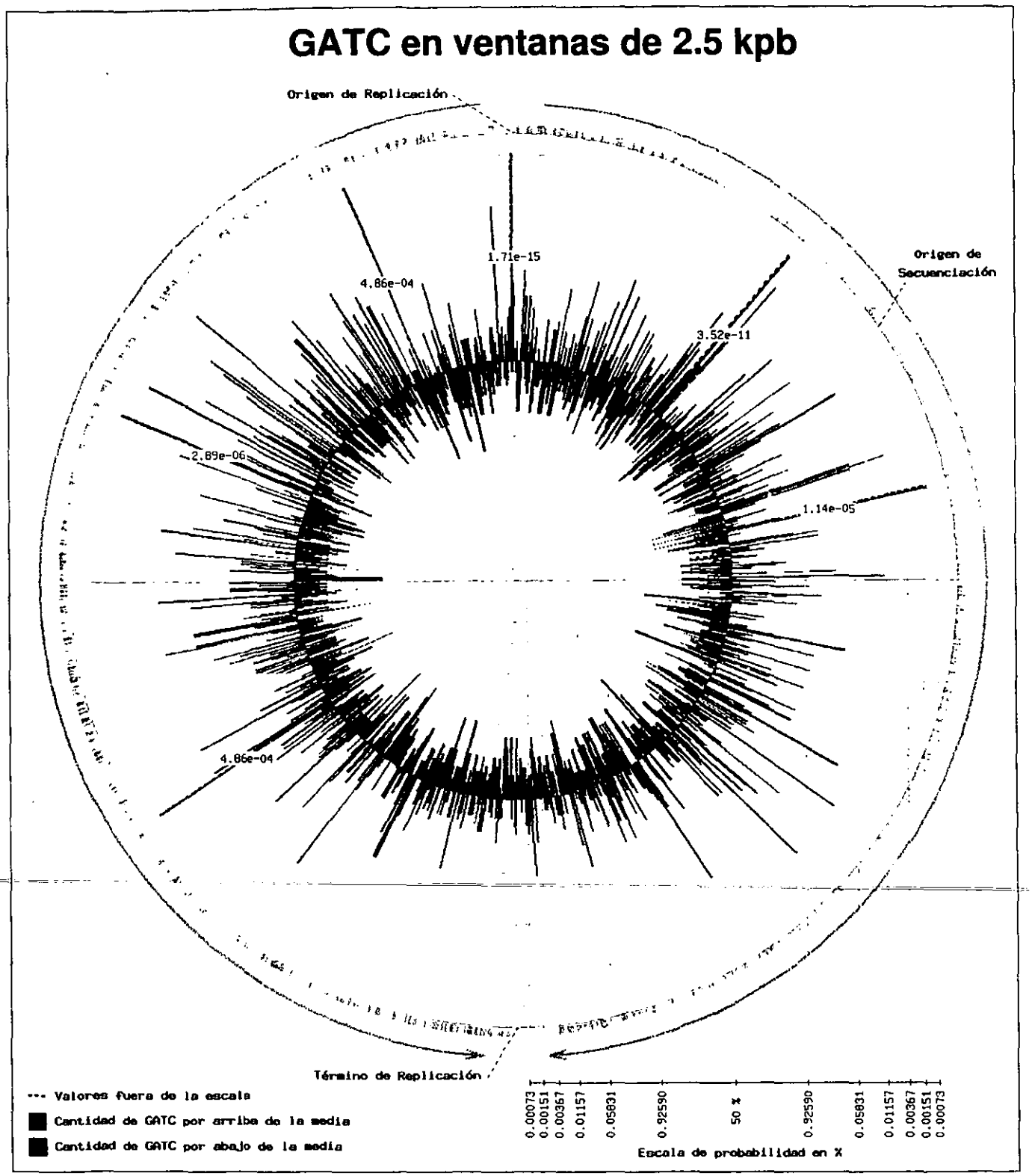


Figura 7. ***Diagrama genómico de las ventanas de 2.5 kpb con GATC***

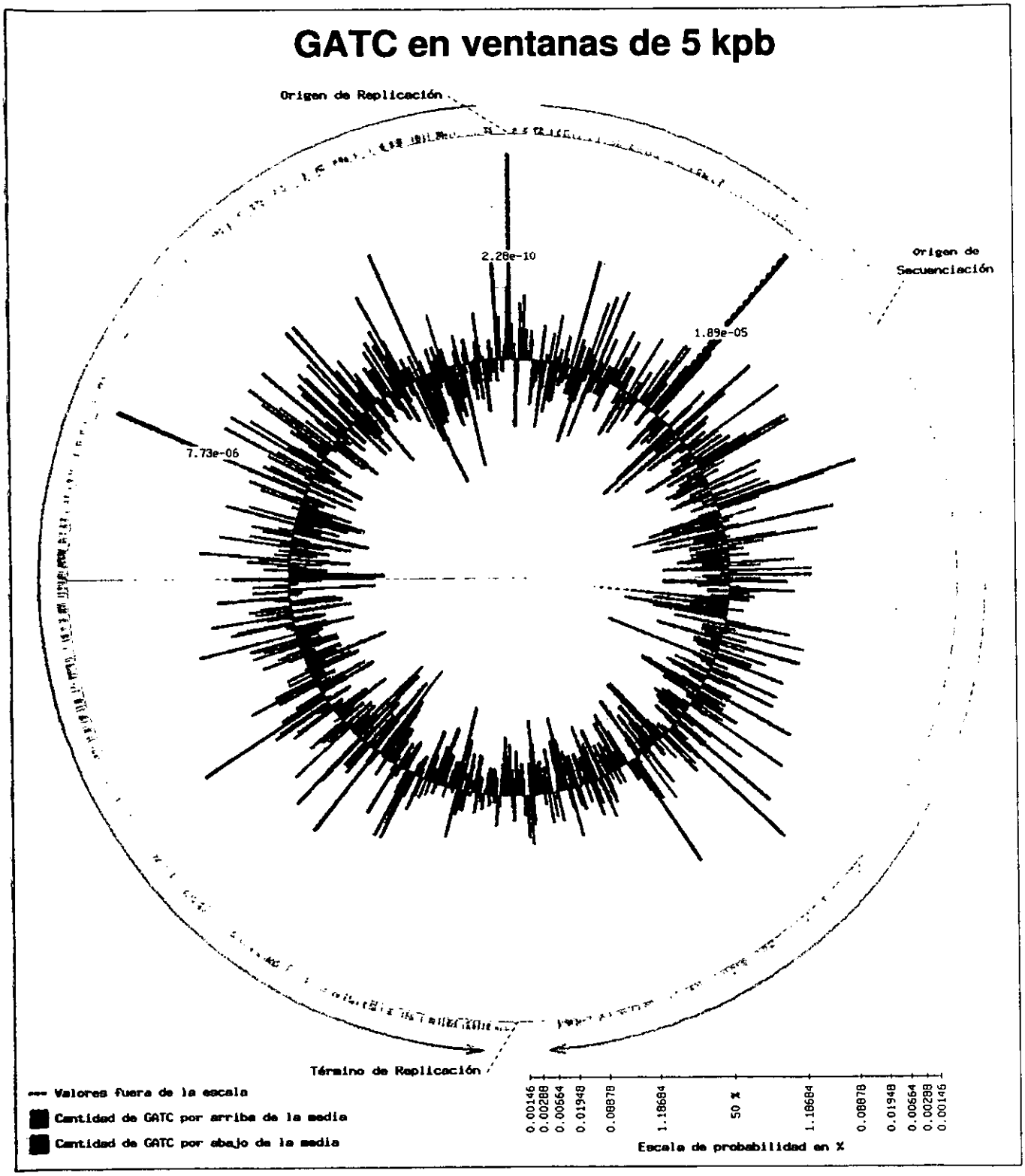


Figura 8. ***Diagrama genómico de las ventanas de 5 kpb con GATC***

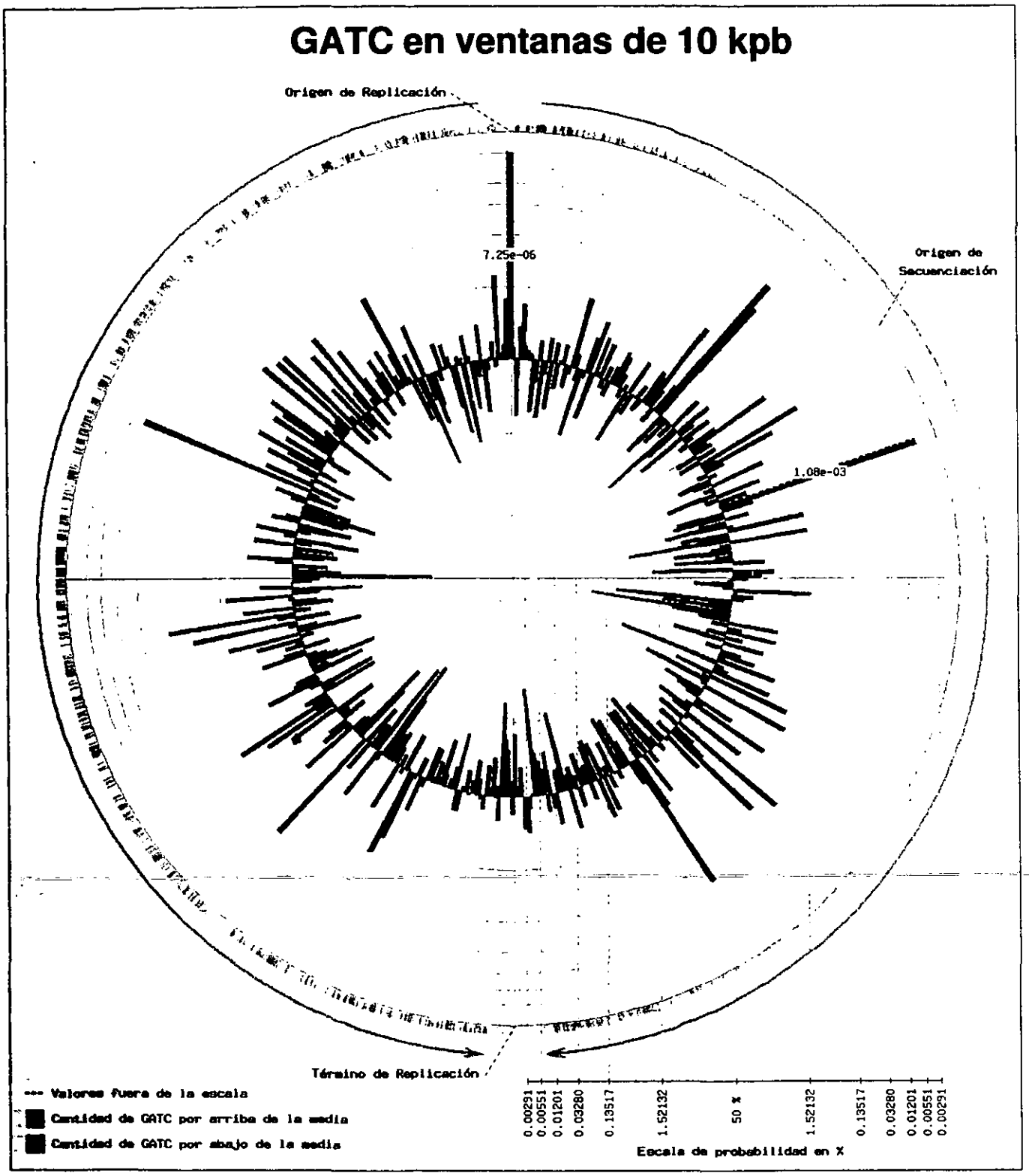


Figura 9. ***Diagrama genómico de las ventanas de 10 kpb con GATC***

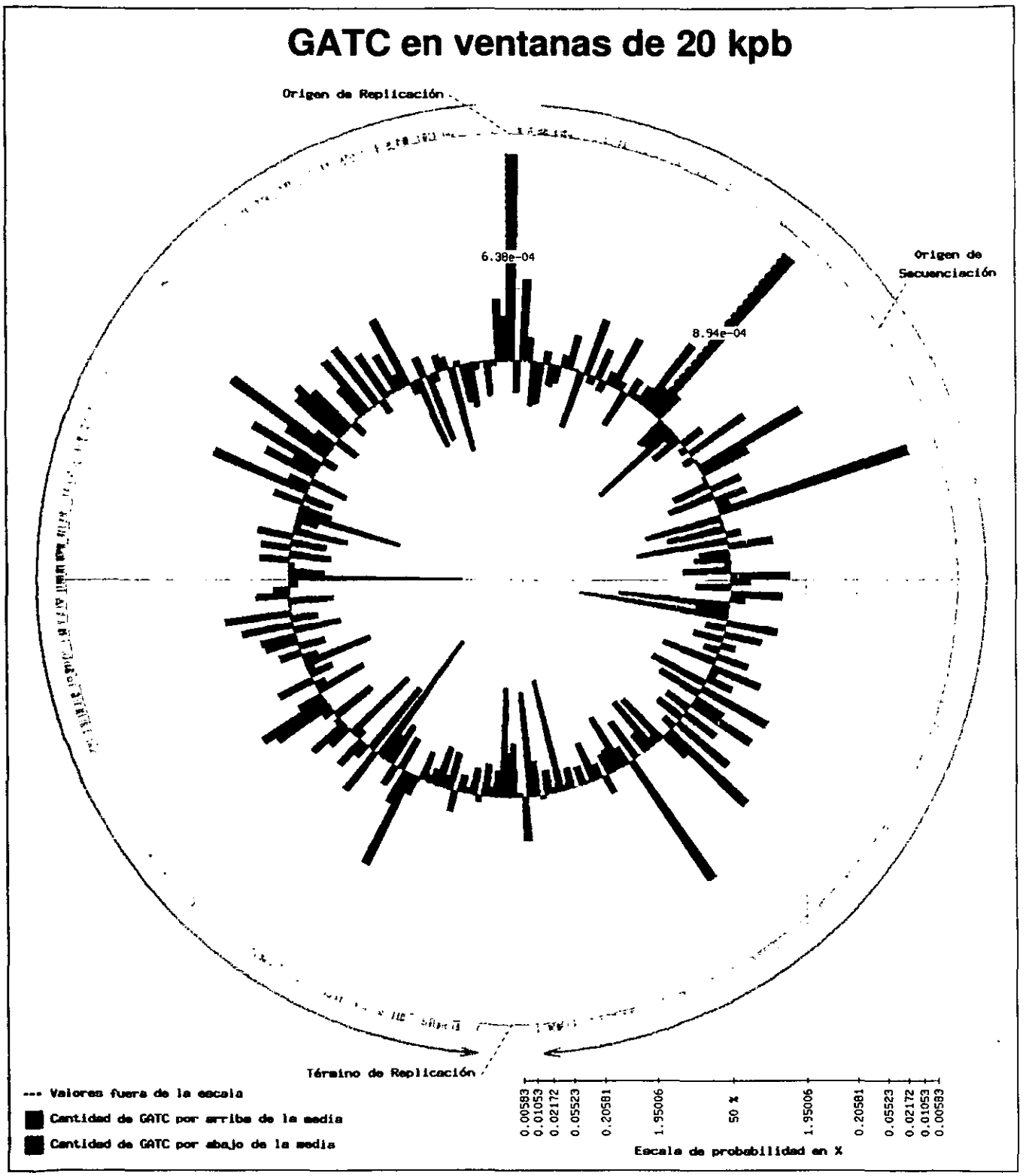


Figura 10. Diagrama genómico de las ventanas de 20 kpb con GATC

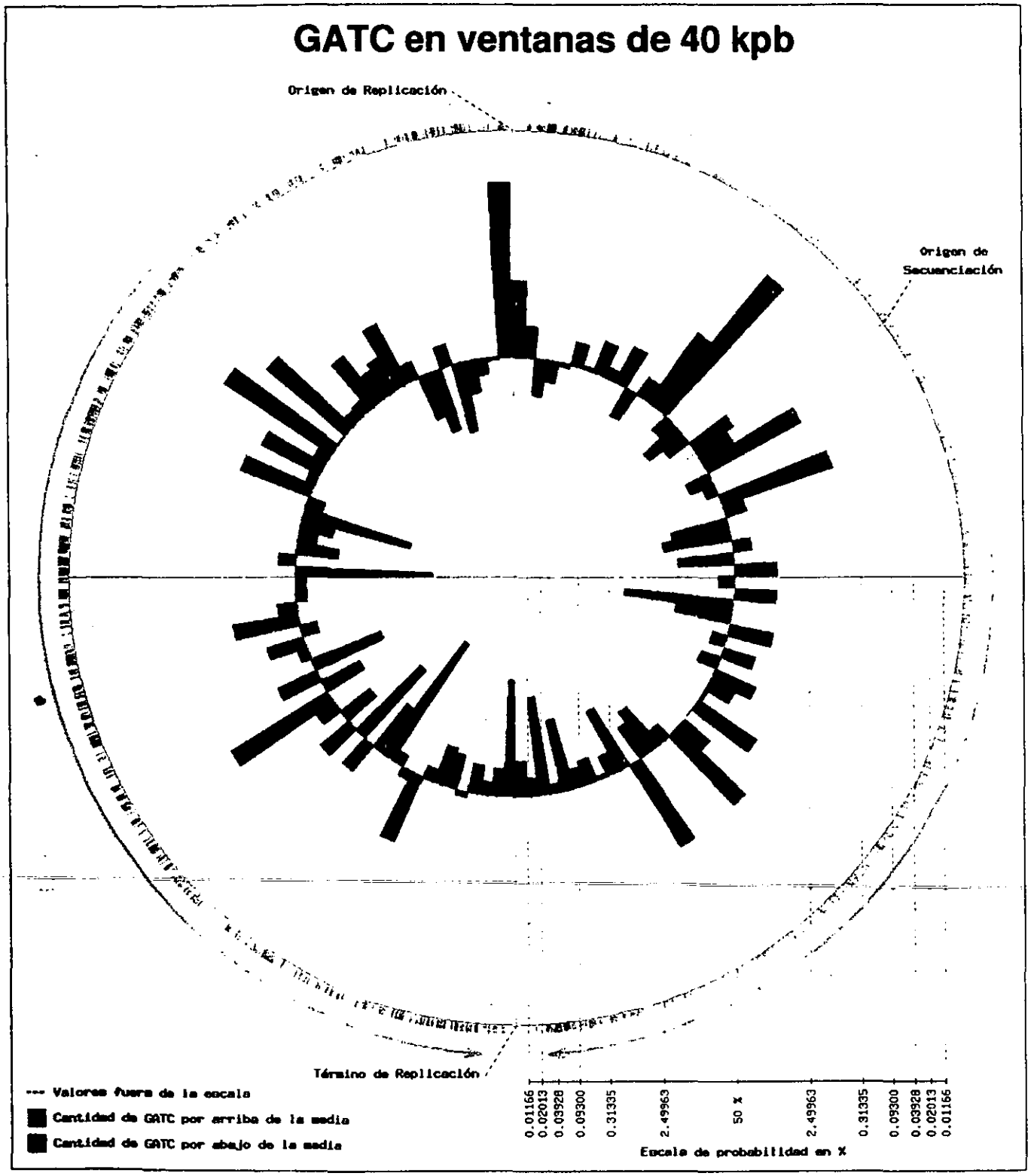


Figura 11. Diagrama genómico de las ventanas de 40 kpb con GATC

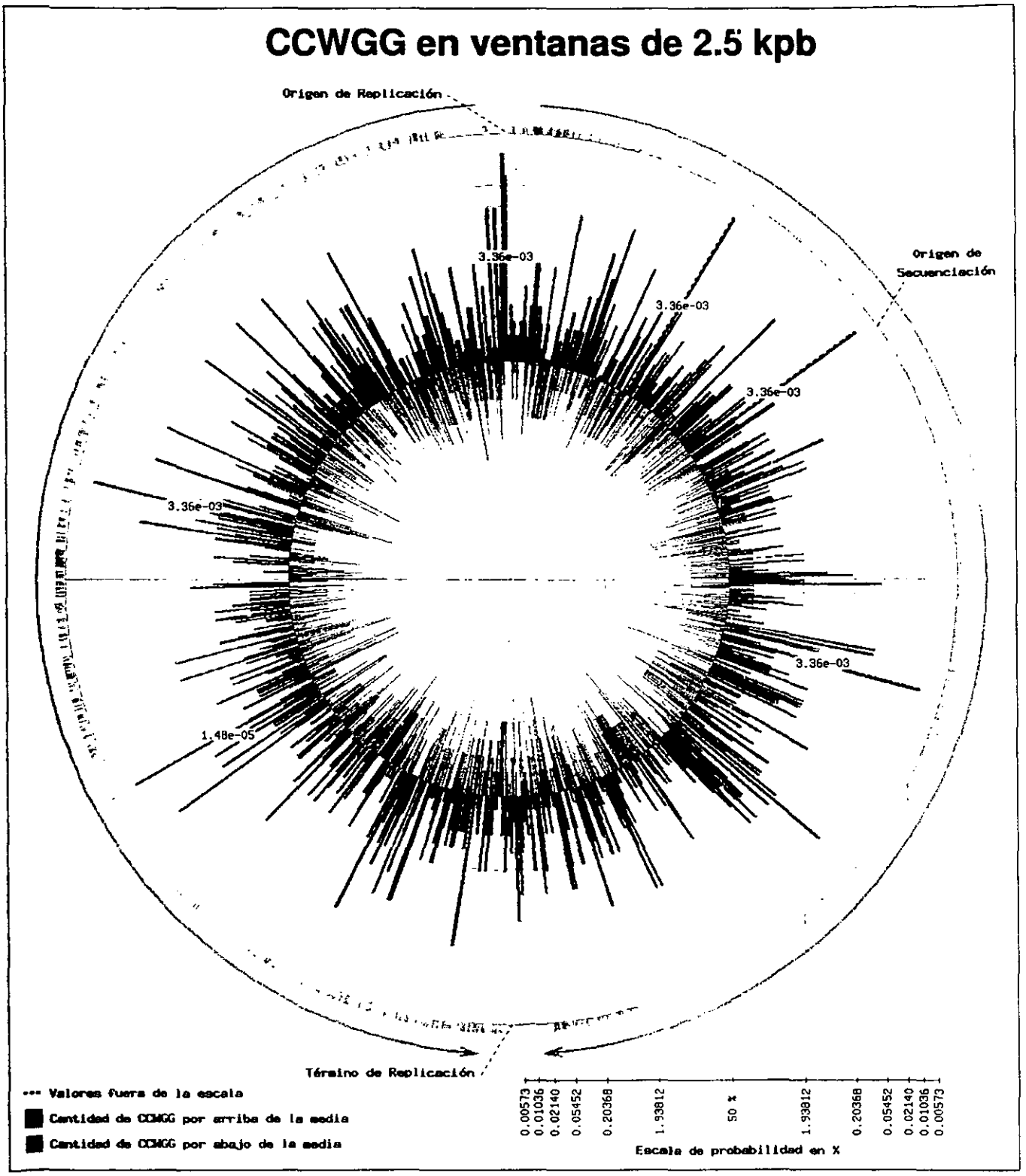


Figura 12. Diagrama genómico de las ventanas de 2.5 kpb con CCWGG

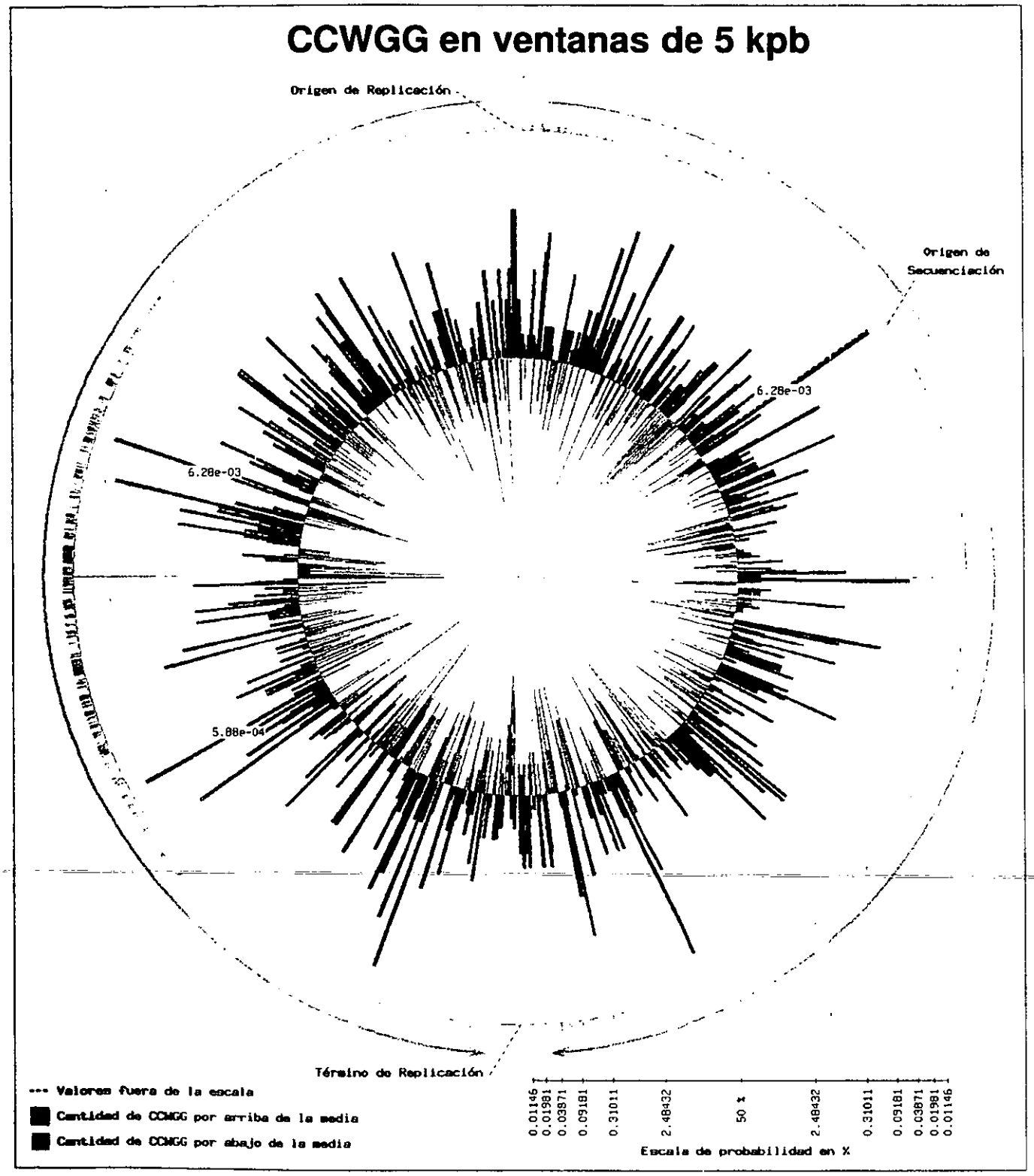


Figura 13. Diagrama genómico de las ventanas de 5 kpb con CCWGG

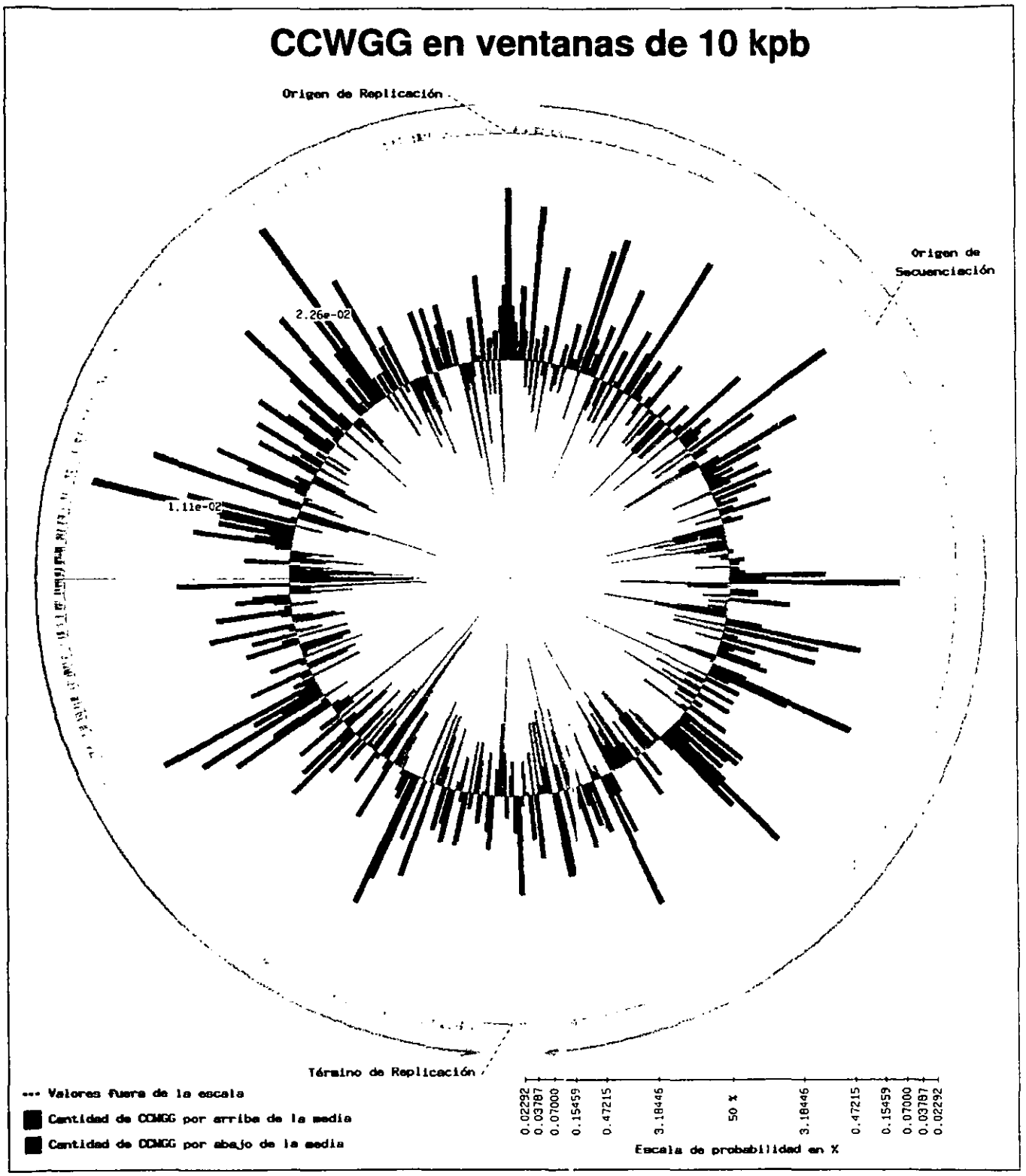


Figura 14. Diagrama genómico de las ventanas de 10 kpb con CCWGG

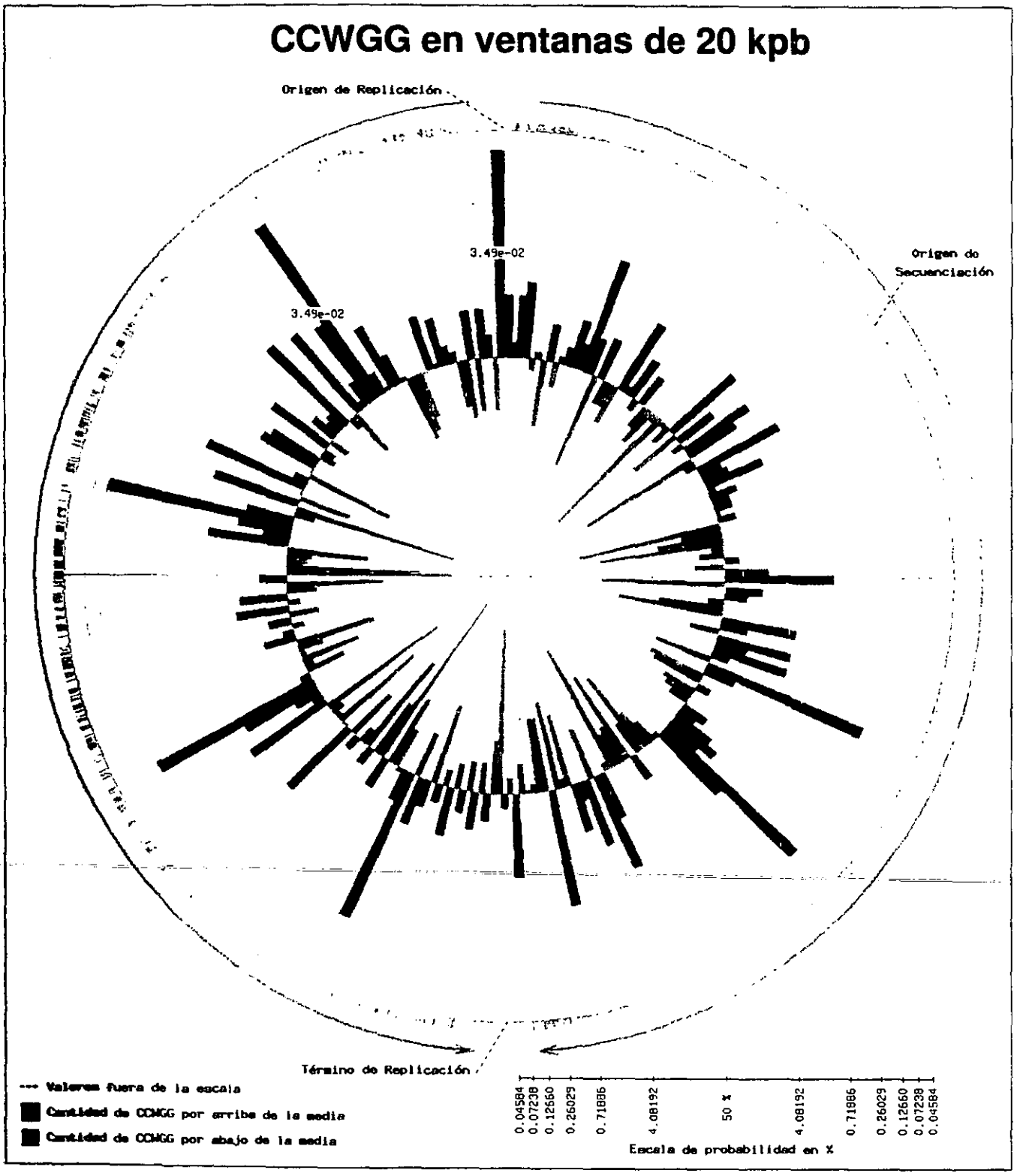


Figura 15. Diagrama genómico de las ventanas de 20 kpb con CCWGG

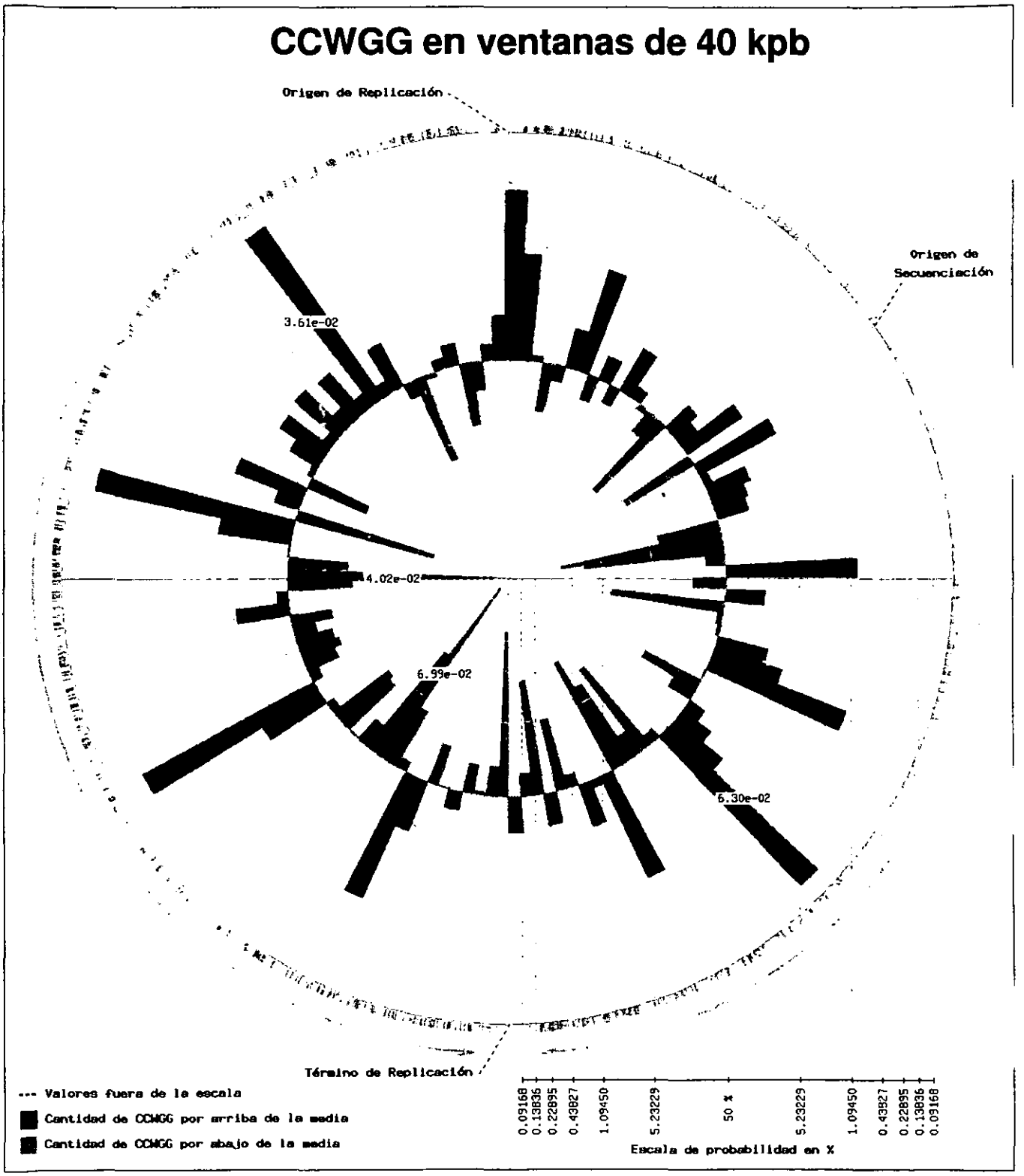


Figura 16. Diagrama genómico de las ventanas de 40 kpb con CCWGG

En su mayor parte, las distintas ventanas parecen ser distribuidas al azar alrededor del cromosoma. Los diagramas para CCWGG ejemplifican esto. Sin embargo, los diagramas genómicos de GATC presentan un detalle inesperado. Las ventanas más significativas tienden a caer en la mitad del cromosoma centrado sobre el origen de replicación y de una manera casi simétrica (Figuras 7 y 8). La más importante es siempre, como era sabido, la ventana que contiene al origen mismo. Como se ha discutido antes, este detalle estructural es bien conocido y se sabe que funciona para mediar la sincronización del inicio de la replicación. Esto ocurre debido a que el DNA hemimetilado generado detrás de las horquillas de replicación de esta región es secuestrado por una fracción de la membrana (Ogden, et al. 1988). Las regiones así capturadas se vuelven inactivas tanto transcripcionalmente como para efectos del inicio de replicación. Así, Dam tiene que competir con la fracción proteica de la membrana por las secuencias presentes alrededor de *oriC* y cuando logra metilarlos completamente, el DNA se desprende y puede volver a activarse. Adicionalmente, la interacción de los orígenes con la membrana había sido interpretado como un mecanismo posible para la segregación de los cromosomas, al ir creciendo la membrana podía ir separando los cromosomas (Ogden, et al. 1988). Sin embargo, al encontrarse que mutantes *dam* al dividirse no producen minicélulas (sin cromosoma) con una mayor frecuencia que las silvestres, se ha terminado por aceptar que la segregación cromosomal ocurre sin problemas en ausencia de Dam (Vinella, et al. 1992). Aunque no tenemos los suficientes datos para invalidar la idea aceptada, la simetría de las ventanas con cantidades significativas de GATC alrededor del origen parecería apoyar la idea de la segregación dependiente de Dam. Si el fenómeno de anclaje a membrana depende meramente de la concentración de sitios GATC, podrían existir otras regiones que se comporten de la misma manera que el origen. Lo que los diagramas genómicos nos muestran son los posibles candidatos a ser regiones anclables a membrana durante la replicación. En base a los resultados obtenidos en esta tesis, propongo que estas interacciones DNA–membrana podrían favorecer o ayudar a que la segregación cromosomal se realice de una manera apropiada. Si éste es el caso, los resultados sugieren que la región cercana a *oriC* no es la única involucrada. Otras regiones ricas en GATC pueden irse anclando a la membrana y contribuyendo a la separación y/o organización de los cromosomas. En cualquier caso, las regiones con frecuencias significativas de sitios Dam pueden verse acopladas al ciclo celular. De la misma manera que ocurre para *oriC* y *dnaA*, cualquier región que se secuestre en membrana quedaría

transcripcionalmente inactiva por un periodo definido del ciclo de la duplicación del DNA.

Los diagramas genómicos de metilaciones combinadas (Figuras 17–21) fueron realizados con la idea de buscar regiones que tuvieran concentraciones significativas (positivas o negativas) de ambos tipos de metilación. Esto para apoyar la hipótesis de que existen regiones cuya tasa de mutagénesis se encuentra determinada principalmente por su patrón de metilación. De esta manera, regiones con un alto contenido de GATC (bien protegidos por el mecanismo de reparación) y bajo contenido de CCWGG (pocas mutaciones por desaminación de citosina metilada) tendrían una tasa baja de mutación. El otro caso bien definido sería el que tiene baja frecuencia de GATC y muchos sitios CCWGG, determinando así una tasa de cambio relativamente alta. Las otras dos posibles combinaciones son menos claras. En el caso de encontrar regiones con frecuencias bajas para ambos tipos de metilación, decidimos clasificarlas como poco mutagénicas. Esta elección se debió a que probablemente tengan la cantidad mínima de GATC para que el mecanismo de corrección funcione (1 sitio cada 2,000 pb) y tienen pocos sitios CCWGG mutagénicos. El último caso es el de ventanas que presentan frecuencias por arriba de la media de ambos sitios. Éstas las consideramos del lado de las mutagénicas ("negativas" en las Figuras 17–21) y requiere de una explicación. Si ocurre una desaminación de citosina metilada (produciendo timina) y la horquilla de replicación se encuentra cercana, los sistemas de reparación (VSR y el dependiente de Dam) competirían por corregir el error. Si la reparación se lleva a cabo dirigida por un sitio GATC hemimetilado, la mutación quedará fijada en el 50% de los casos. Entonces, una región con un contenido alto de CCWGG tiene una mayor probabilidad de contener sitios desaminados al momento de replicarse. Si a esto se le agrega un alto contenido de sitios Dam, la competencia entre los dos mecanismos de reparación podrá ser ganada más fácilmente por el que depende de los sitios GATC, resultando en una tasa de cambio mayor. Debido a que las probabilidades combinadas fueron calculadas meramente como el producto de las probabilidades para cada tipo de metilación por separado, un valor muy significativo para un tipo de metilación puede arrastrar el valor conjunto sin que la cantidad del segundo tipo de metilación sea realmente significativa. Por esta y otras consideraciones que discutiremos en la última parte de los resultados, sólo podemos afirmar que aunque si existen regiones con concentraciones alternadas o equiparables de ambos tipos de metilación, los resultados hasta ahora no son concluyentes.

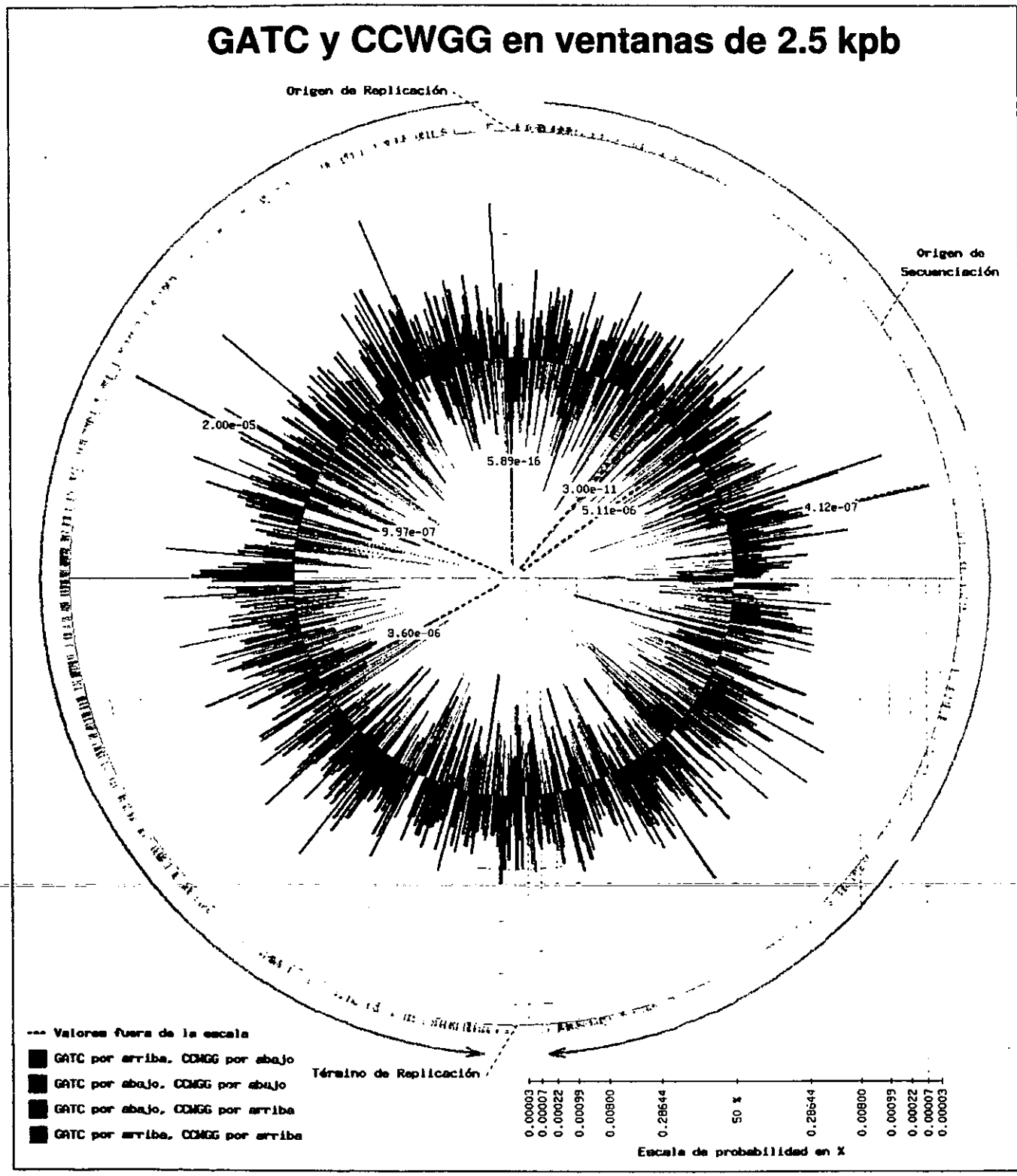


Figura 17. Diagrama genómico de las ventanas de 2.5 kpb con GATC y CCWGG

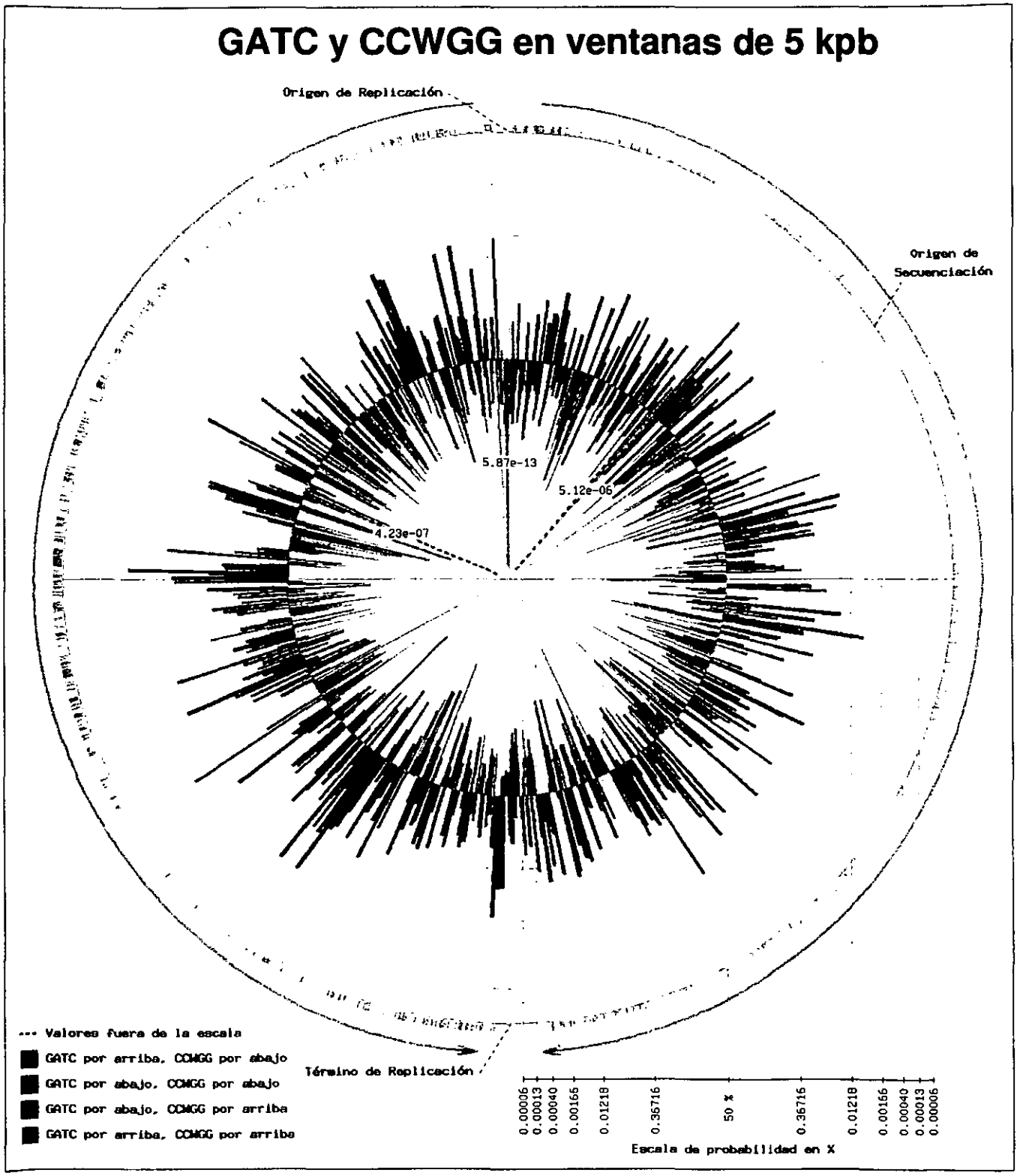


Figura 18. Diagrama genómico de las ventanas de 5 kpb con GATC y CCWGG

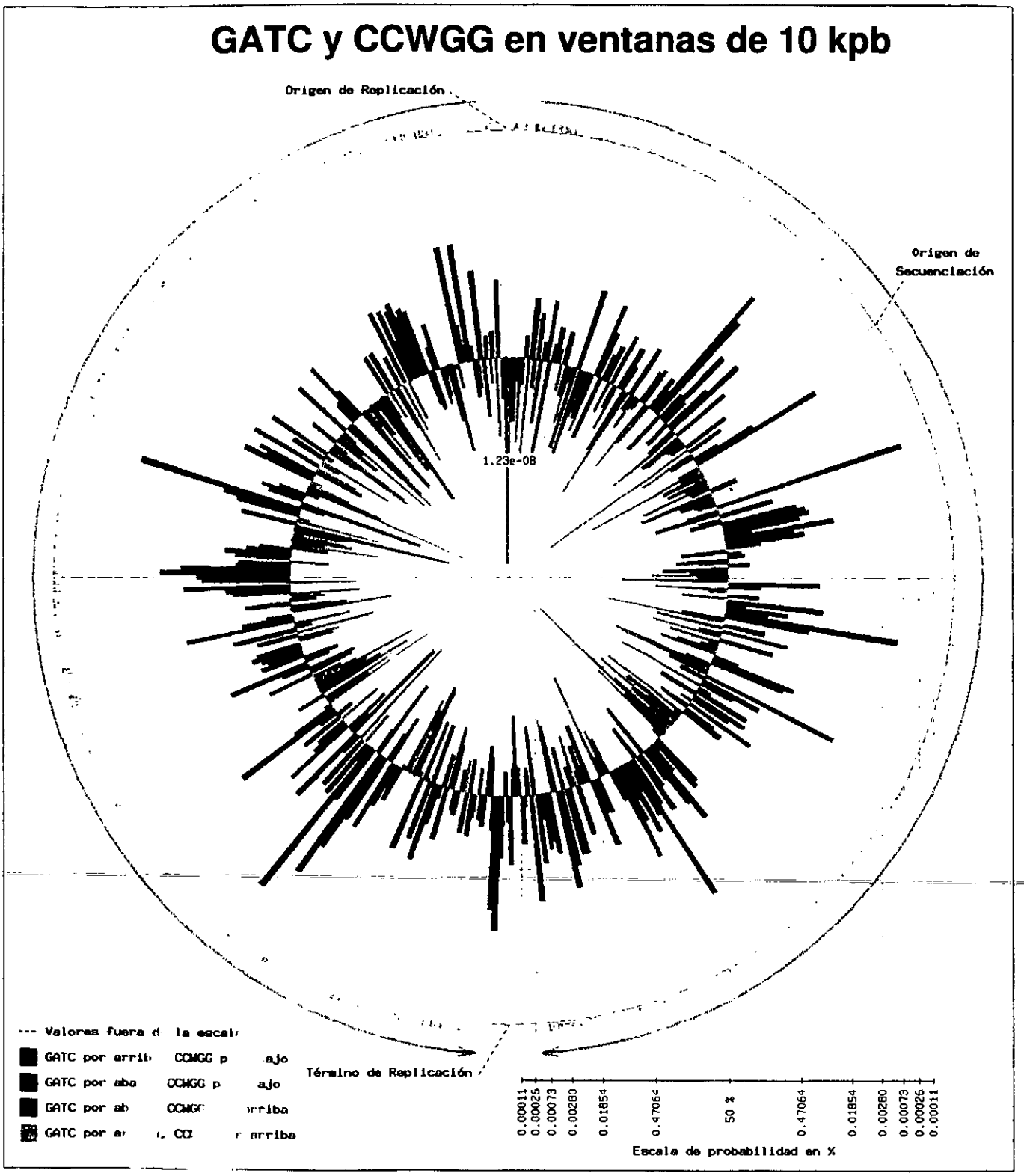


Figura 15. Diagrama genómico de las ventanas de 10 kpb con GATC y CCWGG

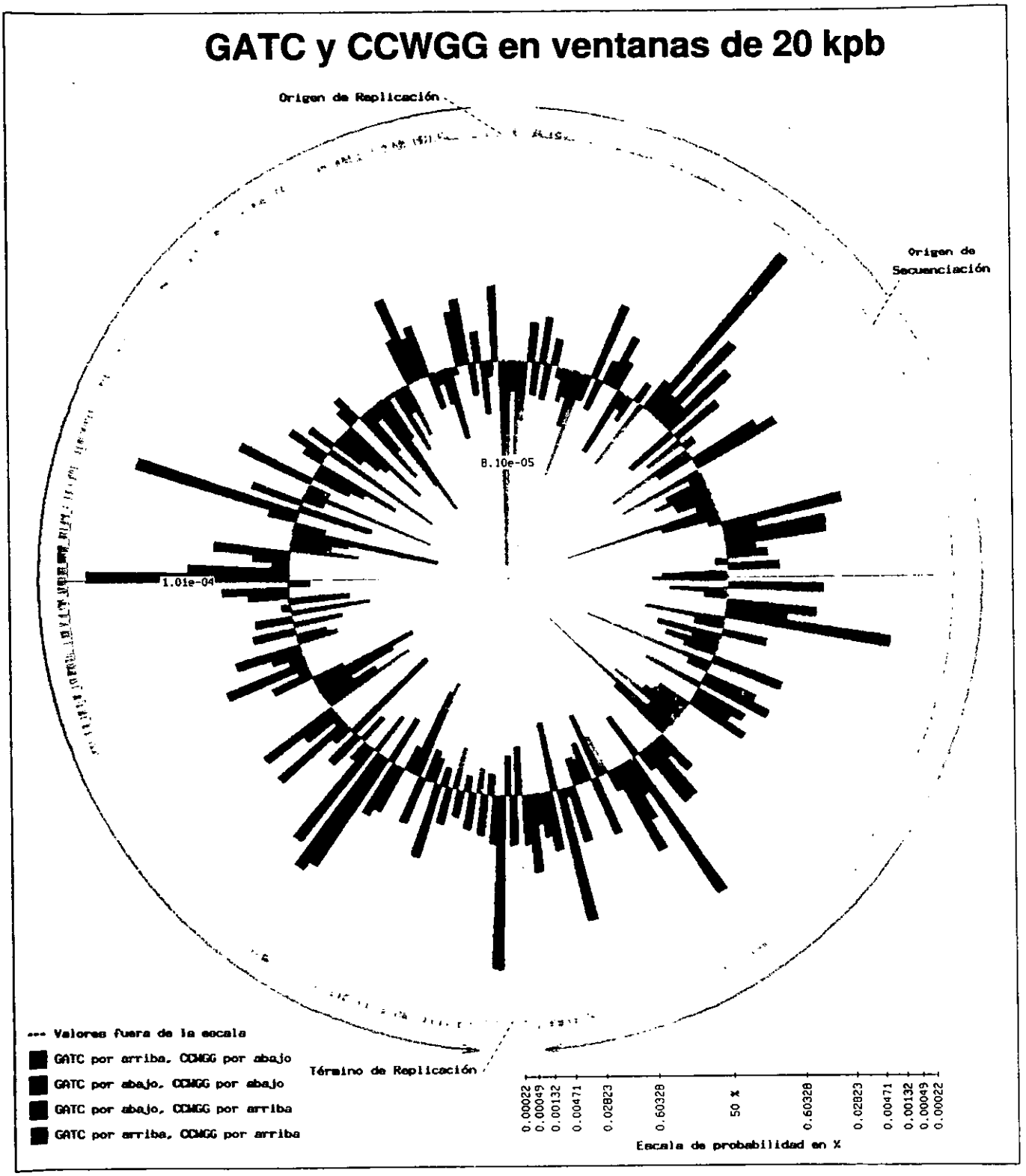


Figura 20. Diagrama genómico de las ventanas de 20 kpb con GATC y CCWGG

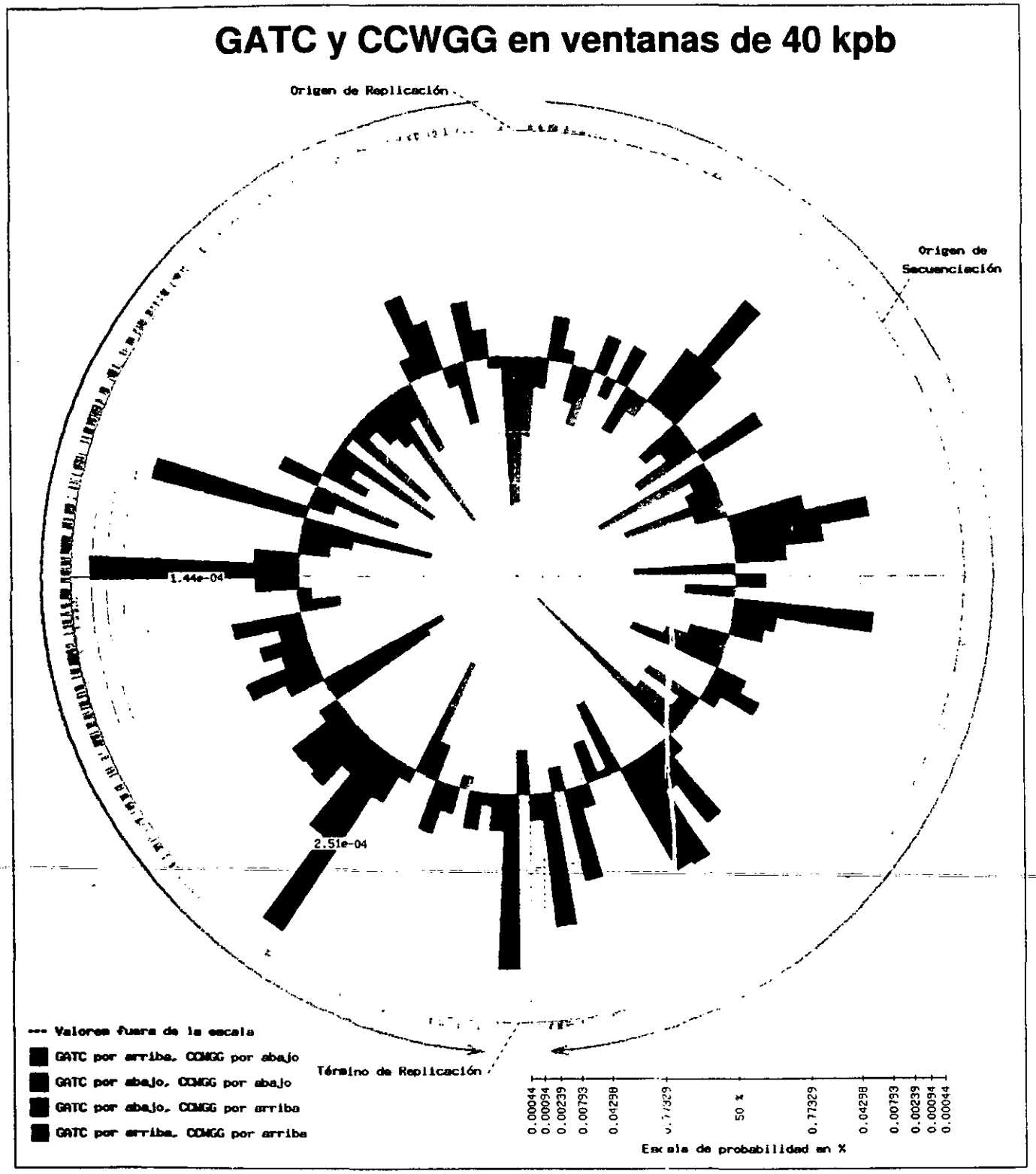


Figura 21. Diagrama genómico de las ventanas de 40 kpb con GATC y CCWGG

Análisis del genoma continuo

Los resultados anteriores nos dan un panorama general de los sitios de metilación en el cromosoma de *E. coli*. Sin embargo, para ciertas cuestiones, un análisis estadístico puede no ser el más conveniente. Por ejemplo, si queremos buscar regiones que no se vean afectadas por la mutagénesis de Dcm, es más significativo buscar las regiones más grandes posibles que no contengan estos sitios de metilación, sin importar si son de un mismo tamaño (como se hace en el análisis por ventanas). Así mismo, regiones que pudieran tener una importancia estructural dada su densidad de sitios de metilación, pudieran ser mucho más pequeñas que aún el tamaño de ventana más pequeño elegido (por ejemplo, la región correspondiente al origen de replicación mide 232 pb). Por este tipo de consideraciones, decidimos contestar ciertas preguntas al nivel del genoma completo, sin división previa alguna. Los datos a obtener, para cada tipo de metilación, fueron los siguientes: i) las regiones más grandes posibles que no tuvieran un sólo sito de metilación (vacíos) y ii) los agrupamientos mayores pero en fragmentos mucho más pequeños que las del análisis por ventanas.

La búsqueda de vacíos produjo resultados inesperados. Hasta ahora, en todos los trabajos enfocados al estudio de los sitios GATC, no se había encontrado una sola región de más de 2,000 pb sin al menos un sitio (Marinus, 1996). Esto parecía perfectamente aceptable, ya que es el tamaño máximo sobre el que puede actuar el sistema de reparación dependiente de Dam (Modrich, 1991). La conclusión era que todo el genoma de *E. coli* se encontraba protegido por este sistema. A lo largo del presente trabajo, diseñamos programas *de novo* para contestar explícitamente cada pregunta (ver Metodología y Programas). Esto nos permitió ser mucho más rigurosos y flexibles que si dependieramos solamente de programas y datos ya disponibles. Con el Programa 14, encontramos no sólo uno, si no veintiún vacíos mayores a 2 kb, el más grande de 4,839 pb (Tabla 3). Las mutaciones en estas regiones tienen una probabilidad baja de ser corregidas por el sistema dependiente de Dam. Aunque estos vacíos sólo constituyan el 1.3% del cromosoma, son muy buenos candidatos para analizar más a fondo su contenido.

Tabla 3. Los mayores 25 vacíos de sitios de metilación			
Vacíos de GATC		*Vacíos de CCWGG*	
Tamaño	*Posición*	*Tamaño*	*Posición*
4839	521307	4140	2753826
4081	3759387	3819	2532885
3937	728527	3779	3794081
3835	3616646	3192	1473177
3510	2466591	3191	2764516
3369	2069370	3128	2453296
3268	2766105	3091	4465982
2781	1065698	3017	2065021
2762	2778338	2967	2278764
2705	2188008	2912	1389749
2580	2650900	2852	779561
2490	1801510	2815	2420316
2368	2074302	2808	3893174
2311	236753	2780	1103218
2310	285763	2764	2780830
2299	3706928	2641	2525224
2292	864899	2585	2989047
2119	262405	2570	3801091
2105	2970708	2564	3125944
2059	2078166	2562	1579707
2019	1426786	2559	1814739
1928	1415457	2512	728864
1924	3259626	2508	3687815
1912	32115	2506	471550
1893	3381767	2486	4446162

Curiosamente, aunque el promedio o valor esperado de sitios CCWGG es menor que para GATC, los vacíos de este tipo de sitio que encontramos nunca fueron tan grandes (Tabla 3). El vacío mayor de sitios Dcm fue de 4,140 pb, cuando no hubiera sido sorprendente, estadísticamente, encontrar alguno que duplicara el tamaño de los de Dam. Aunque no encontramos regiones de tamaño extremo, existen muchos más vacíos de tamaño intermedio

para Dcm que para Dam. Los vacíos mayores de CCWGG encontrados también son buenos candidatos para un estudio más profundo.

Para estudiar los agrupamientos más significativos, decidimos tomar un tamaño mucho más cercano al del origen de replicación (232 pb). Esto debido a que es el ejemplo más claro documentado de una función biológica dependiente de la densidad de los sitios. El tamaño que terminamos escogiendo fue aquél tal que se esperaba encontrar al azar y en promedio un sólo sitio de metilación. Este tamaño fue tomado de las frecuencias observadas en el genoma completo (Tabla 1). Para GATC equivale a una región de 242 pb y para CCWGG, 385 pb. Este tamaño nos permitió resaltar perfectamente al fragmento que contiene el origen de replicación, con 12 sitios GATC (12 veces por encima de lo esperado). Con el Programa 15 obtuvimos para ambos tipos de metilación los 100 agrupamientos más significativos. La misma tendencia de que la presencia de sitios Dam sea más extrema fue observada. Existen 9 agrupamientos de GATC que se encuentran 9 veces, y 23 que están 8 veces, por arriba de lo esperado, aparte del que contiene el origen. Los agrupamientos de CCWGG más grandes están solamente 7 veces por encima de lo esperado, existiendo 4 de éstos, seguidos por 33 agrupamientos 6 veces por encima de lo esperado. Estos datos se muestran en la Tabla 4.

Tabla 4. Los mayores agrupamientos de sitios de metilación			
Agrupamientos de GATC		*Agrupamientos de CCWGG*	
Cantidad	*Posición*	*Cantidad*	*Posición*
12	3923350	7	112100
9	201458	7	777804
9	337248	7	2926999
9	2565355	7	3007583
9	2574771	6	224392
9	2639203	6	456040
9	2953020	6	708427
9	3062135	6	746577
9	3342899	6	1374623
9	3387759	6	1716265
8	60891	6	1863732
8	211243	6	2030689
8	353579	6	2237011

Tabla 4. Los mayores agrupamientos de sitios de metilación			
8	693311	6	2332572
8	2162338	6	2368275
8	2610366	6	2391198
8	2644184	6	2539173
8	2720219	6	2603943
8	3059996	6	2728189
8	3114954	6	3009440
8	3204970	6	3074078
8	3328425	6	3075902
8	3406058	6	3223082
8	3755681	6	3227555
8	3880442	6	3260346
8	3881487	6	3425411
8	4132494	6	3475827
8	4157616	6	3696089
8	4300321	6	3844577
8	4312352	6	3913335
8	4451257	6	3940052
8	4463190	6	4033741
8	4597854	6	4164859
		6	4206346
		6	4335781
		6	4445272
		6	4572415

Los resultados de los vacíos y agrupamientos más significativos presentes en el genoma de *E. coli* se encuentran visualmente en la Figura 22. Como se puede ver, la mayor parte de los agrupamientos de GATC se encuentran en la mitad del cromosoma que contiene al origen. Esto también apoya la idea de que intervienen en la segregación del cromosoma. También significa que los genes que posiblemente se acoplen transcripcionalmente al ciclo celular, debdio a su contenido de sitios Dam, predominan en esta mitad; a diferencia de la que contiene el término de la replicación. Los vacíos, sin embargo, se encuentran distribuídos de una manera bastante homogénea por todo el cromosoma.

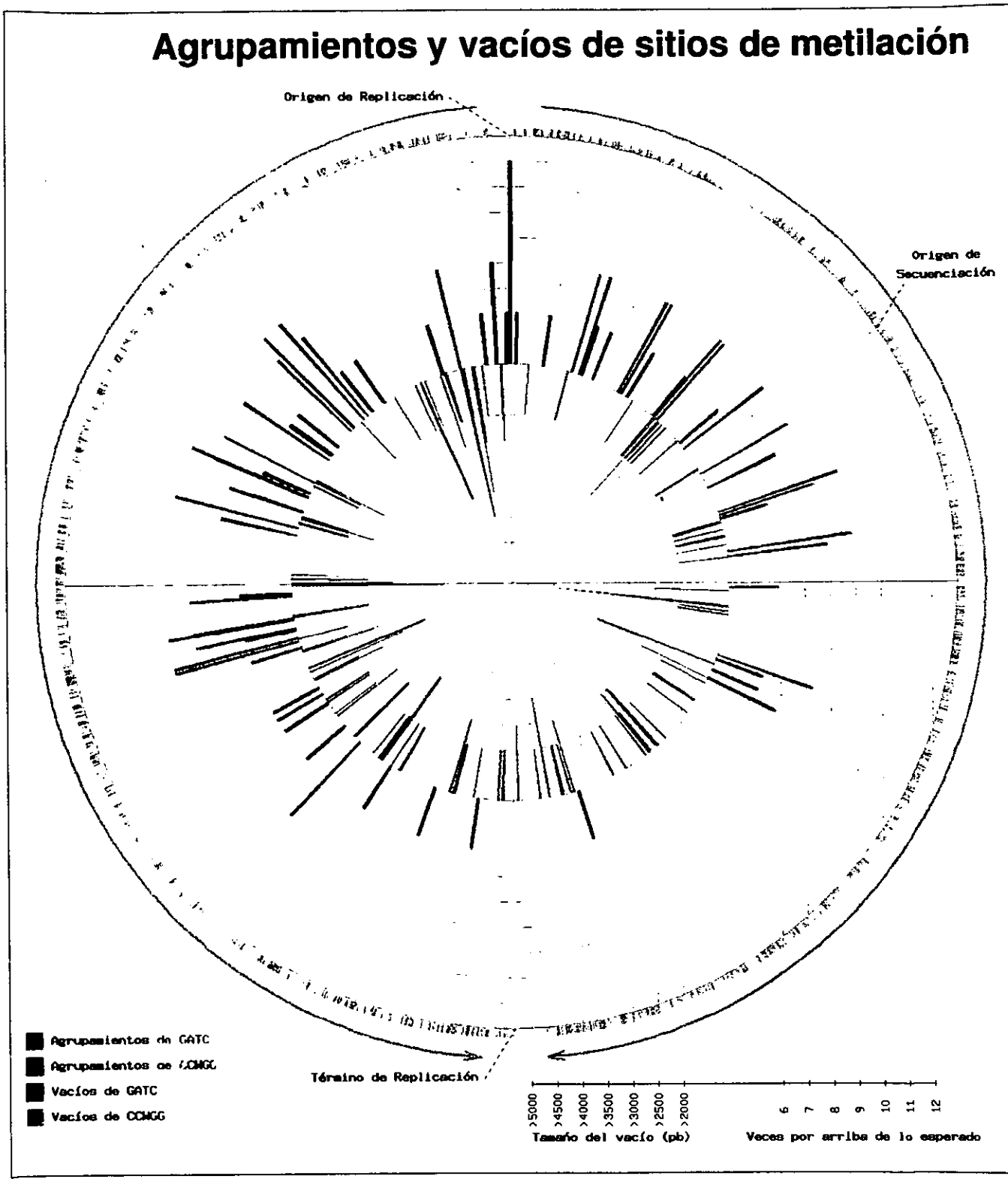


Figura 22. ***Diagrama genómico con los vacíos y agrupamientos más significativos de cada tipo de metilación***

Análisis funcional

Reconsiderando la parte de "posible implicación biológica" del título de esta tesis, debemos tomar en cuenta que la unidad funcional para la mayoría de los fenómenos biológicos es el gen, y no una región abstracta del genoma. Tenemos qúe observar donde caen los sitios de metilación en este contexto, no solamente la cantidad de ellos. Además, una hipótesis que realmente se esperaba poder confirmar o rechazar era la de la mutagénesis dependiente de los sitios de metilación. El trabajo de Gómez-Eichelmann y Ramírez-Santos intentó contestar esta cuestión, pero al no contar con la suficiente cantidad de información (en este caso secuencia), sólo pudieron encontrar un gen que podría presentar una mayor tasa de mutagénesis por su contenido de sitios de metilación (Gómez-Eichelmann y Ramírez-Santos, 1993).

Dos cuestiones generales se contestaron primero. Por un lado necesitábamos saber la frecuencia en que los dos tipos de metilación caían en tres regiones funcionales del genoma; esto es DNA codificante, sólo transcrito, e intergénico (ver Objetivos). Por otro lado, no todas las bases de las regiones codificantes son equivalentes. La segunda base de un codón se encuentra mucho más comprometida con la identidad del aminoácido correspondiente que la primera o tercera base, aunque la tercera basè es la menos comprometida de las tres. Así, también debemos de tomar en cuenta en qué posición del codón caen las bases metilables, especialmente las citosinas metilables por Dcm, ya que dependiendo de esto pueden ser más o menos mutagénicas.

De los tres tipos de regiones funcionales del genoma, se espera que la más conservada o estable sea la codificante. De esta manera, se espera que si realmente los sitios GATC logran una protección deberán encontrarse con una mayor frecuencia en esta región que en aquellas intergénicas o solamente transcritas, donde pudieran no ser tan importantes. De igual manera, los sitios CCWGG, si es necesario evitar las mutaciones que provocan, deberían encontrarse disminuidos especialmente en la región codificante. Como se puede ver en la Figura 23, éste es precisamente el caso.

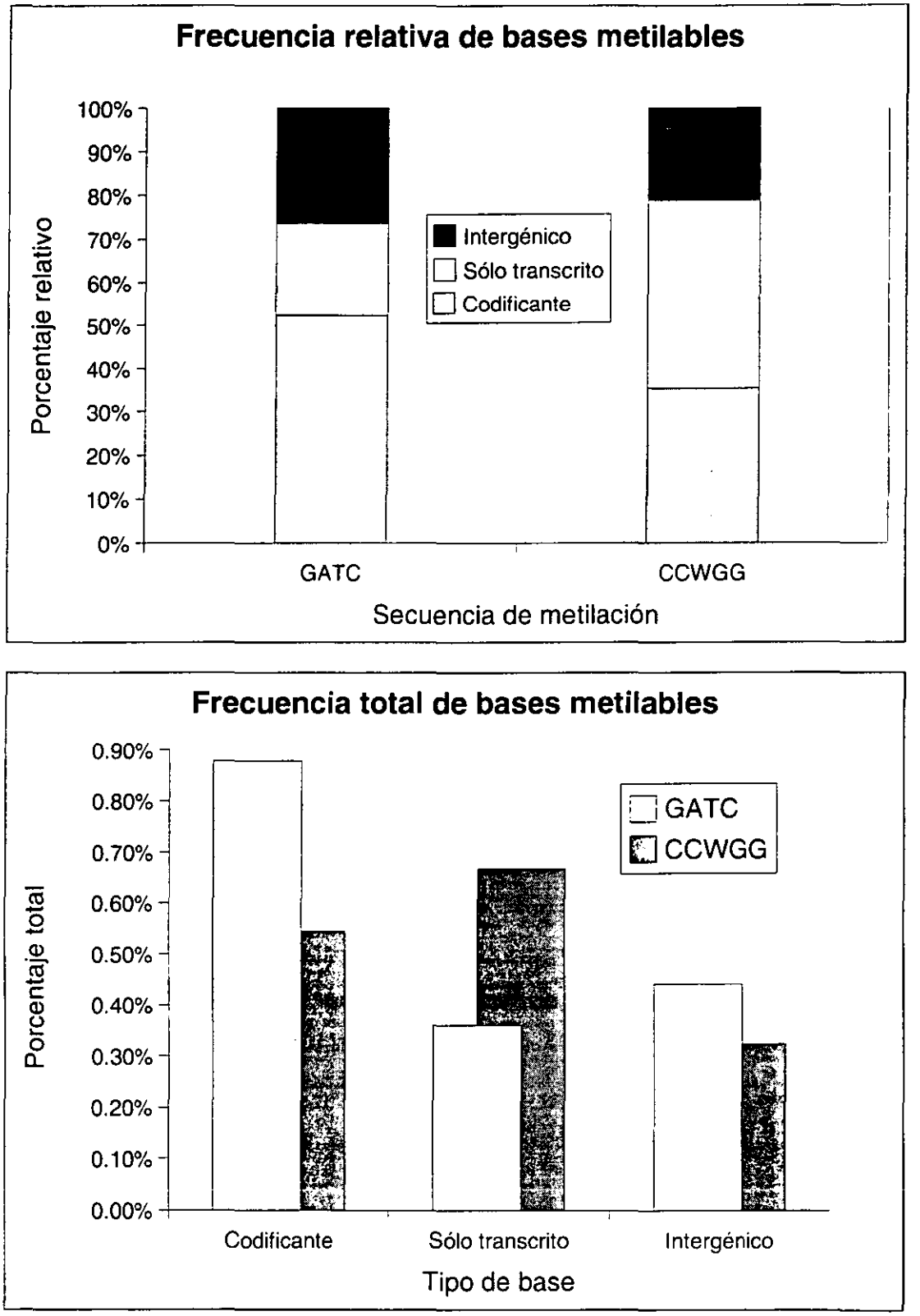


Figura 23. Frecuencia de bases metilables en distintas regiones del genoma

En la región codificante se observa una mayor frecuencia de adeninas metilables que de citosinas metilables. Lo curioso, es que en la región del genoma que solamente se transcribe, ocurre justo lo contrario, existen más citosinas que adeninas modificables. Esta región está formada por genes de los tRNAs y RNAs ribosomales. Una cosa que tenemos que tomar en cuenta es que mientras que GATC contiene una fracción equitativa de A–T y G–C, CCWGG contiene cuatro veces r G–C que A–T. Por ésto, si dos regiones tienen contenidos de G–C diferentes, al azar espera que tengan distinto contenido de CCWGG. La Tabla 5 muestra los porcentajes de G–C para cada uno de los tipos de región aquí mencionados.

Tabla 5. Porcentaje de G–C en los tres tipos de región	
Región	***Porcentaje de G–C***
Intergénico	42.32%
Codificante	51.85%
Sólo transcrito	54.87%

Esta desviación en el porcentaje de G–C que se observa impartiría una tendencia en la cantidad de CCWGG en las distintas regiones. De hecho, es exactamente la misma tendencia que se observa en la Figura 23 (se ve más claramente en la gráfica inferior de la figura) con la mayor representación en la región transcrita, intermedia en la codificante y la menor de las tres en la región intergénica. En vista que GATC contiene la misma cantidad de A–T que de C–G esta tendencia no puede actuar sobre la frecuencia de estos sitios.

Si realmente ha existido una presión por eliminar a las citosinas metilables de las posiciones mutagénicas, esto debe de poder ser fácilmente observable en las regiones codificantes. En vista a que la posición más comprometida de un codón es la segunda, ésta debe tener la menor frecuencia de citosinas modificadas, y la tercera, al ser la posición más laxa podrá tener la mayor concentración. En la Figura 24 se ve precisamente este comportamiento para CCWGG y lo contrario para GATC. En este caso también debemos tomar en cuenta la desviación que puede existir para el porcentaje de G–C en cada posición de los codones. En la Tabla 6 se muestran estos porcentajes para cada posición.

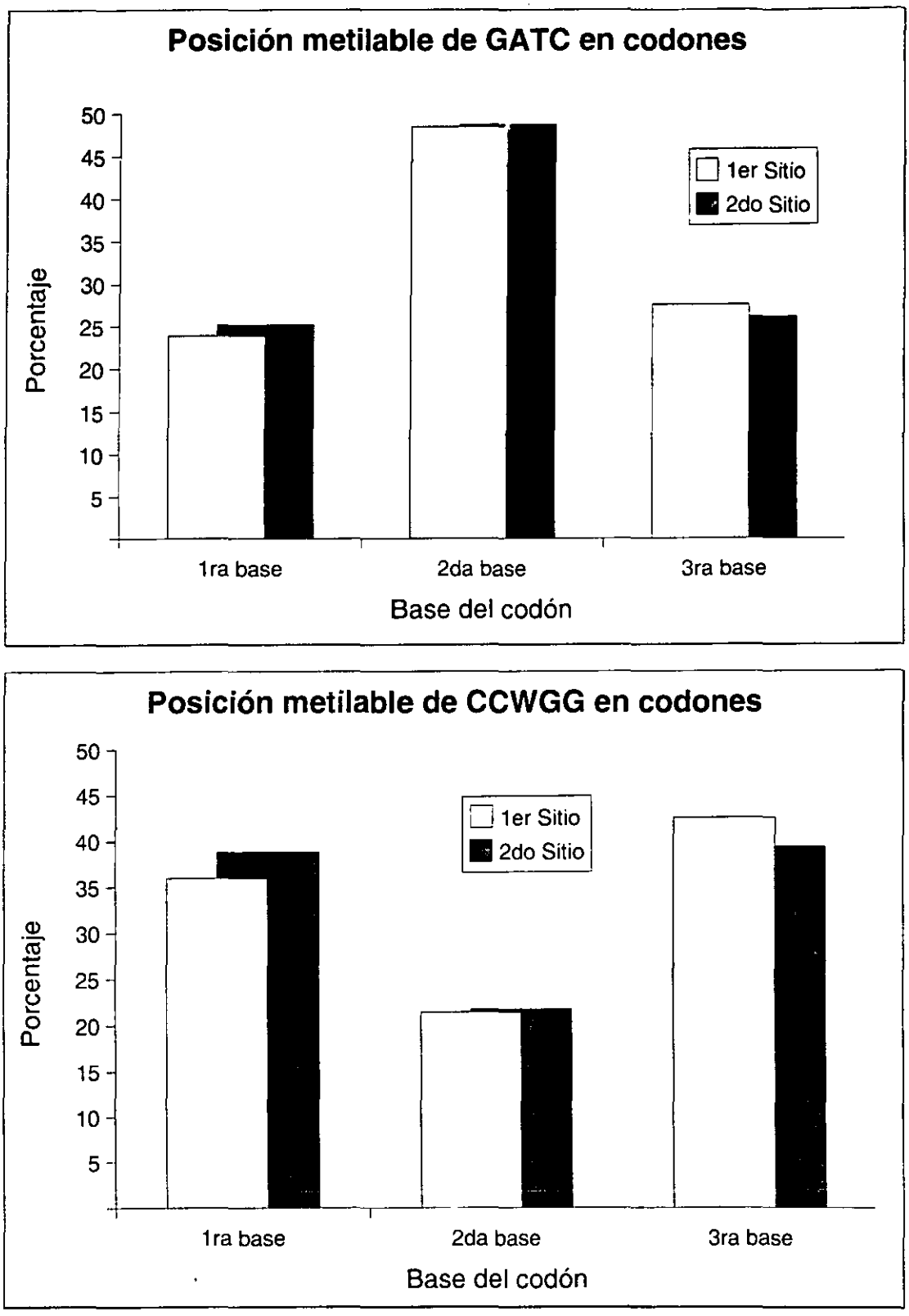


Figura 24. Posición de bases metilables en codones

Tabla 6. Porcentaje de G–C en los codones	
Posición en codón	***Porcentaje de G–C***
1ra base	58.87%
2da base	40.72%
3ra base	55.89%

Considerando que para la secuencia GATC la posición metilable es una adenina y que para CCWGG es una citosina, estos porcentajes se asemejan bastante a la forma de las gráficas de la Figura 24. Aunque el contenido de G–C podría parecer una causa de los resultados presentes en esta figura, realmente se trata de un problema circular. No podemos descartar la posibilidad de que la desviación del porcentaje de G–C observado en cada posición de los codones fue en parte impuesto por una selección actuando al nivel de la citosina metilada.

Una última manera de corroborar la hipótesis fue seleccionar dos grupos de genes. El primer grupo, de genes esenciales, debería ser un compartimiento muy estable, con poca necesidad de cambiar y por lo tanto debería tener una tasa intrínseca de mutación relativamente baja. El segundo grupo es mucho más laxo, y está compuesto por genes que se sabe que no son esenciales. Se espera que estos no tengan tanto compromiso de estabilidad y que puedan tener una tasa intrínseca de mutación por lo menos mayor al primer grupo. En la Figura 25 se puede ver, en primer lugar, la dispersión de todos los genes traducidos de *E. coli*, y en segundo lugar, la dispersión específicamente de los genes esenciales y no esenciales. Aquí se esperaba que los genes esenciales se distinguieran de los que no lo fueran, quedando en la región inferior y derecha (la menos mutagénica, de acuerdo a nuestras consideraciones). Contrario a nuestra espectativa, no existe tal distinción. Esto podría parecer una indicación de que la hipótesis debe ser rechazada. Sin embargo, cabe considerar dos cuestiones antes de hacerlo. Por un lado, quizá el grupo elegido no fue el más apropiado. La hipótesis dicta que debería existir un grupo de genes que dado su contenido de sitios de metilación sea menos variable que el resto. Por ello escogimos un grupo de genes para los que existe evidencia experimental que son esenciales, pero esto no significa necesariamente que son mutagénicamente estables.

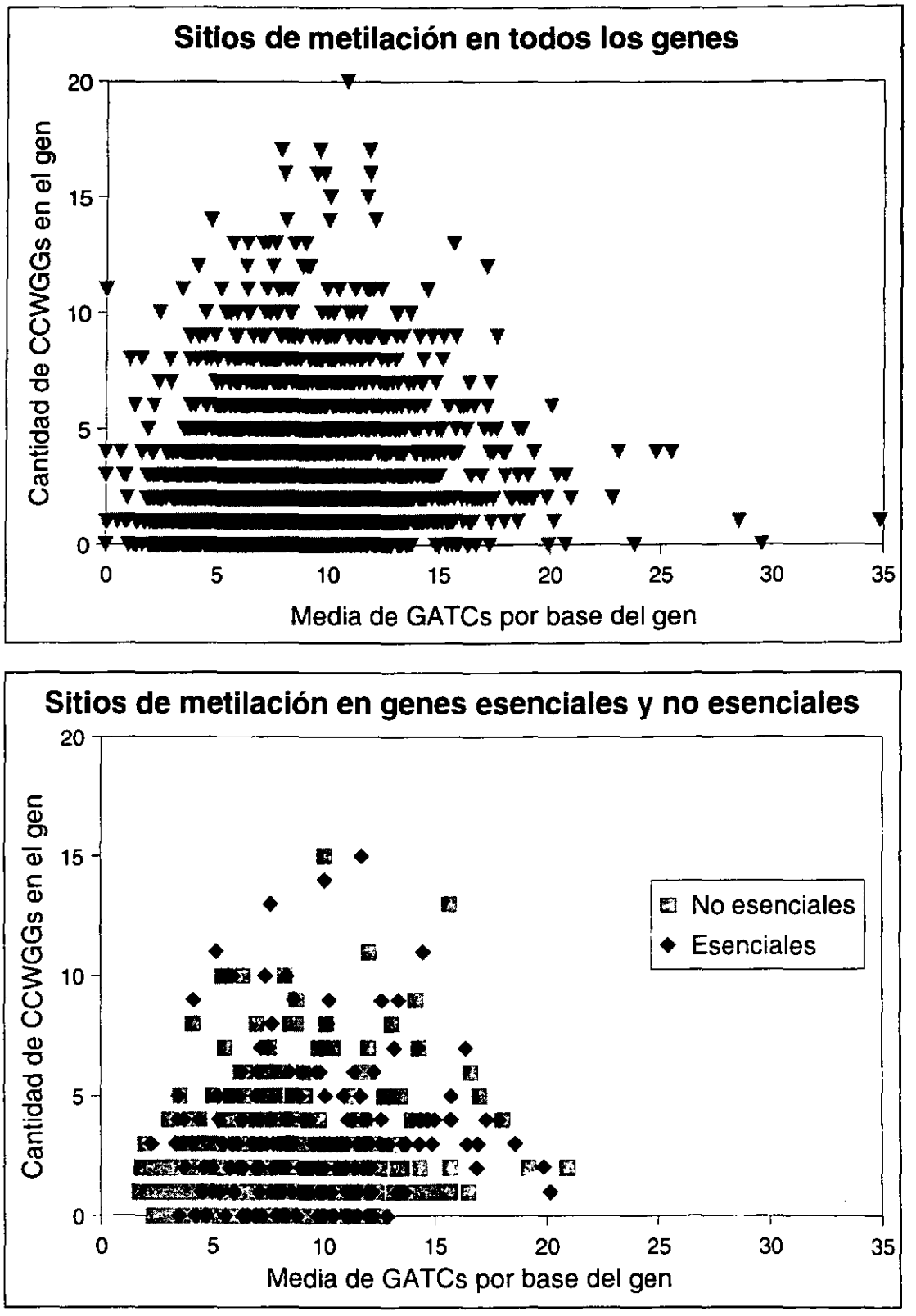


Figura 25. Dispersión de genes dado su contenido de sitios de metilación

Lo que convendría encontrar, es un grupo de genes que se sepa que han cambiado muy poco a través del tiempo, por lo menos en los clados que presentan actualmente los sistemas de metilación Dam y Dcm. El mismo estudio podría hacerse con este grupo de genes y los resultados tendrían mayor repercusión sobre la hipótesis. La otra cuestión es que quizá estemos considerando mal la utilidad biológica de los sitios Dcm. Podría ser que los efectos más importantes ocurren en regiones reguladoras y no en regiones codificantes, actuando como un modulador de la expresión. Entonces convendría hacer un estudio donde se consideren los promotores de los genes, comparándolos con las regiones aquí estudiadas. Consideramos estas y otras cuestiones en la sección de Perspectivas.

CONCLUSIONES

- En *Escherichia coli* K–12, las frecuencias de los sitios de metilación, GATC y CCWGG, se encuentran por debajo de lo esperado estadísticamente. La comparación con el análisis markoviano indica que existe una selección negativa para la presencia de ambos sitios. La presión en contra de la secuencia CCWGG es la mayor.

- La cantidad de sitios de metilación por ventana se encuentra distribuido de acuerdo a una función normal.

- Las ventanas y los agrupamientos más sobresalientes por su contenido de GATC se encuentran distribuidas principalmente alrededor del origen de replicación. Esto apoya la idea de una segregación cromosomal ayudada por estas regiones.

- La ventana más significativa por su contenido de GATC es la que incluye al origen de replicación. Esto coincide con los resultados publicados por otros grupos.

- A diferencia de lo que se creía, el cromosoma de *E. coli* no puede ser protegido eficientemente en su totalidad por el mecanismo dependiente de Dam. Existen 21 regiones de más de 2000 pb sin un sólo sitio GATC.

- Las regiones, de mayor a menor frecuencia de sitios GATC, son: i) codificante, ii) intergénico, iii) sólo transcrito. La región más protegida por GATC, es por lo tanto la codificante.

- Las regiones, de mayor a menor frecuencia de sitios CCWGG, son: i) sólo transcrito, ii) codificante, iii) intergénico.

- Las posiciones en los codones con adenina metilable, de mayor a menor frecuencia, son: 2da > 3ra > 1ra.

- Las posiciones en los codones con citosina metilable, de mayor a menor frecuencia, son: 3ra > 1ra > 2da. Esto puede ayudar a que la desaminación de la citosina metilada no ocasione un cambio de aminoácido.

- Un grupo de genes, para los que existe evidencia experimental de que son esenciales, no pudo ser distinguido por su contenido de sitios Dam y Dcm.

PERSPECTIVAS

Aún falta mucho trabajo por realizar para concluir realmente la discusión sobre los sistemas Dam y Dcm en *E. coli*, especialmente en cuanto a las funciones de Dcm y la relación de ambos con la tasa intrínseca de mutación. Hace falta realizar el análisis de los genes contenidos en ciertas regiones señaladas en este trabajo. Los genes presentes en ventanas significativas o más importantemente en agrupamientos de GATC deberían estar transcripcionalmente acoplados a la replicación del cromosoma. Dentro de este tema, la relación de los agrupamientos de sitios Dam con los procesos involucrados en la división celular y la segregación cromosomal, aún no ha sido descrita completamente. Va a hacer falta mucho trabajo experimental si se desea esclarecer. Los genes contenidos en los vacíos de GATC también deberán ser estudiados experimentalmente, para ver si realmente su tasa de cambio es mayor al promedio. También se necesitaría estudiar así las regiones significativas por su contenido de CCWGG, cuando se tenga un mayor conocimiento de la función de Dcm en la fisiología de *E. coli*, estos genes pueden resultar interesantes. Como una exploración inicial he decidido incluir un apéndice (ver Genes en Regiones Seleccionadas) con los genes que se encuentran dentro de algunas de las regiones que esta tesis ha resaltado: los vacíos y los agrupamientos de sitios de metilación. Estas tablas facilitarán el trabajo posterior que se realice en esta área. El análisis a fondo de estos datos es material para un trabajo futuro, pero por ahora basta mencionar que contienen detalles de interés. Es sugestivo que los cuatro vacíos más importantes de GATC se encuentran ocupados por cuatro de los cinco elementos *rhs* (Recombination Hot Spot). No pareciera que esto fuera fortuito, requiere de una explicación. Curiosamente, la región donde se lleva a cabo el inicio de la replicación no se encuentra adentro del agrupamiento mayor, si no a 290 pb de distancia y no contiene gen alguno. Esto parece razonable si consideramos que una región con tal contenido de sitios GATC difícilmente puede tener la flexibilidad necesaria para contener mucha más información. Todos estos datos deberán ser analizados con detenimiento para poder ser explicados.

El estudio de las frecuencias y posiciones de sitios Dcm puede realizarse de una manera más completa. Debido a que los sistemas Dcm y VSP actúan en sentidos contrarios,

intercambiando CCWGG y sus derivados desaminados, los sitios sobre los que VSP actúa directamente deberían entrar en un futuro estudio. Además, por simplicidad aquí solamente consideramos la posición del codón en que caían las citosinas metilables. Lo estrictamente correcto sería considerar cada caso y contar como mutagénicas aquellas que realmente cambian el aminoácido, sin importar si están en primera, segunda o tercera posición del codón. Esto podría realizarse en un trabajo posterior, aunque no se espera realmente que cambie mucho los resultados (ya que la generalidad es que la segunda posición es la más comprometida, y la tercera la menos, con la identidad del aminoácido).

La hipótesis de que existan regiones cuya tasa de mutación depende principalmente de los sitios de metilación sigue siendo factible. Para redondear este trabajo, y tomar una decisión conclusiva sobre esta hipótesis, requerimos de más datos. En lo que concierne a la Figura 25, se podrían escoger mejores grupos para el análisis, como se mencionó al final de la sección de Resultados y Discusión. Un grupo de genes para el que exista evidencia evolutiva que haya cambiado poco, sería el candidato ideal a comparar contra un grupo que se sepa que haya cambiado mucho. Otra manera de atacar el problema sería tomar distintas regiones de la gráfica superior de la Figura 25 y buscar características que tuvieran en común los genes así agrupados.

Una excelente manera de validar todas las propuestas planteadas en esta tesis y en cualquier trabajo posterior sería realizar los mismos experimentos pero con otras bacterias que no tengan uno u otro sistema de metilación. Esto sería de vital importancia especialmente para las conclusiones acerca de Dcm. Por ejemplo, pudimos haber descartado datos que no parecían significativos (especialmente junto a los datos de Dam), pero por poco importantes que parecían en este estudio, si no aparecieran en una cepa naturalmente Dcm$^-$, implicaría que sí es importante para la bacteria que cuenta con este sistema. Existe justo esta cepa, *E. coli* B no tiene el sistema Dcm, sin embargo su secuencia aún no ha sido publicada. Aunque ésta sería el candidato ideal para el estudio comparativo, de las secuencias de genomas completos con los que contamos en la actualidad, podríamos utilizar el de *Haemophilus influenzae* o el de *Vibrio cholerae* ya que son filogenéticamente cercanos a *E. coli* (gama proteobacterias) y al parecer tienen Dam, pero no Dcm.

Aunque esta tesis no haya sido del todo conclusiva, es un ejemplo del tipo de cuestiones que se pueden retomar, o las nuevas ideas con las que se puede trabajar en esta nueva y fascinante era de la genómica.

BIBLIOGRAFÍA

Barras F, Marinus MG. **1988**. Arrangement of Dam methylation sites (GATC) in the *Escherichia coli* chromosome. Nucleic Acids Research. 16:9821–9838.

Barras F, Marinus MG. **1989**. The Great GATC: DNA methylation in *E. coli*. TIG. 5:139–143.

Benson DA, Karsch–Mizrachi I, Lipman DJ, Ostell J, Rapp BA, Wheeler DL. **2000**. GenBank. Nucleic Acids Research. 28:15–18.

Bertani G, Weigle JJ. **1953**. Host controlled variation in bacterial viruses. J. Bacteriol. 65:113–121.

Bjork GR, Ericson JU, Gustafsson CE, Hagerwall TG, Jonsson YH, Wikstrom PM. **1987**. Transfer RNA modification. Ann. Rev. Biochem. 56:263–287.

Blattner FR, Plunkett G, Bloch CA, Perna NT, Burland V, Riley M, Collado–Vides J, Glasner JD, Rode CK, Mayhew GF, Gregor J, Davis NW, Kirkpatrick HA, Goeden MA, Rose DJ, Mau B, Shao Y. **1997**. The Complete Genome Sequence of *Escherichia coli* K–12. Science. 277:1453–1462.

Boye E, Marinus MG, Loner–Olesen A. **1992**. Quantitation of Dam methyltransferase in *Escherichia coli*. J. Bacteriol. 174:1682–1685.

Bramhill D, Kornberg A. **1988**. A model for initiation at origins of DNA replication. Cell. 54:915–918.

Campbell JL, Kleckner N. **1990** *E. coli oriC* and the *dnaA* gene promotor are sequestered from *dam* methyltransferase following passage of the chromosomal replication fork. Cell. 62:967–979.

Ciria R, Merino E. **2001**. Comunicación personal. Instituto de Biotecnología, UNAM.

Crothers DM, Haran TE, Nadeau JG. **1990**. Intrinsically bent DNA. J. Biol. Chem. 265:7093–7099.

Duncan BK, Miller JH. **1980**. Mutagenic deamination of cytosine residues in DNA. Nature. 287:560–563.

Ehrlich M, Gama–Sosa MA, Carreira LH, Ljungdahl LG, Kuo KC, Gehrke CW. **1985**. DNA methylation in thermophilic bacteria: N^4–methylcytosine, 5–methylcytosine, and N^6–methyladenine. Nucleic Acids Research. 13:1399–1412.

Franklin RE, Gosling RG. **1953**. Molecular structure of nucleic acids. Nature. 171:740.

Gómez–Eichelmann MC, Ramírez–Santos J. **1993**. Methylated Cytosine at Dcm (CCA/TGG) Sites in *Escherichia coli*: Possible Function and Evolutionary Implications. Journal of Molecular Evolution. 37:11–24.

Haberman A. **1974**. The bacteriophage P1 restriction endonuclease. J. Mol. Biol. 89:545–563.

Hagerman PJ. **1990**. Pyrimidine 5–methyl groups influence the magnitude of DNA curvature. Biochemistry. 29:1980–1983.

Hotchkiss RD. **1948**. The quantitative separation of purines, pyrimidines, and nucleosides by paper chromatography. J. Biol. Chem. 168:315–332.

Janulaitis A, Petrūsyte M, Maneliene Z, Klimasauskas S, Butkus V. **1992**. Purification and properties of the *Eco*57I restriction endonuclease and methylase prototypes of a new class (type IV). Nucleic Acids Res. 20:6043–6049.

Lundblad V, Kleckner N. **1984**. Mismatch repair mutations of *Escherichia coli* K–12 enhance transposition excision. Genetics. 109:3–19.

Luria SE, Human ML. **1952**. A nonhereditary, host–induced variation of bacterial viruses. J. Bacteriol. 64:557–569.

Marinus MG. **1987**. DNA methylation in *Escherichia coli*. Annu. Rev. Genet. 21:113–131.

Marinus MG. **1996**. Chapter 53: Methylation of DNA. *Escherichia coli* and *Salmonella* Cellular and Molecular Biology. Eds. Neidhardt FC, et al. ASM Press. 2nd Edition. 782–791.

Marinus MG, Morris NR. **1974**. Biological function for 6–methyladenine residues in the DNA of *Escherichia coli* K–12. J. Mol. Biol. 85:309–322.

Marinus MG, Poteete A, Arraj JA. **1984**. Correlation of DNA adenine methylase activity with spontaneous mutability in *Escherichia coli* K–12. Gene. 28:123–125.

Meisel A, Bickle TA, Krüger DH, Schroeder C. **1992**. Type III restriction enzymes need two inversely oriented recognition sites for DNA cleavage. Nature. 355:467–469.

Modrich P. **1991**. Mechanisms and biological effects of mismatch repair. Annu. Rev. Genet. 25:229–253.

Murchie AIH, Lilley DMJ. **1989**. Base methylation and local DNA helix stability: effect on the kinetics of cruciform extrusion. J. Mol. Biol. 205:593–602.

Noyer–Weidner M, Trautner TA. **1993**. Methylation of DNA in Prokaryotes. EXS. 64:39–108.

Ogden GB, Pratt MJ, Schaechter M. **1988**. The replicative origin of the *E. coli* chromosome binds to cell membranes only when hemimethylated. Cell. 54:127–135.

Oka A, Sugimoto K, Takanami M, Hirota Y. **1980**. Replication origin of the *Escherichia coli* K–12 chromosome: the size and structure of the minimum DNA segment carrying the information for autonomous replication. Mol. Gen. Genet. 178:9–20.

Parker B, Marinus MG. **1992**. Repair of DNA heteroduplexes containing small heterologous sequences in *Escherichia coli.* Proc. Natl. Acad. Sci. USA. 89:1730–1734.

Phillips GJ, Arnold J & Ivarie R. **1987**. Mono– through hexanucleotide composition of the *Escherichia coli* genome: a Markov chain analysis. Nucleic Acids Research. 15:2611–2626.

Redaschi N, Bickle TA. **1996**. Chapter 52: DNA Restriction and Modification Systems. *Escherichia coli* and *Salmonella* Cellular and Molecular Biology. Eds. Neidhardt FC, et al. ASM Press. 2nd Edition. 773–778.

Revel HR, Luria SE. **1970**. DNA-glucosylation in T-even phage: genetic determination and role in phage-host interaction. Annu. Rev. Genet. 4:177-192.

Roberts D, Hoopes BC, McClure W, Kleckner N. **1985**. IS*10* transposition is regulated by DNA adenine methylation. Cell. 43:117-130.

Roberts RJ, Macelis D. **1994**. REBASE- restriction enzymes and methylases. Nucleic Acids Res. 22:3628-3639.

Sancar A, Sancar GB. **1988**. DNA repair enzymes. Annu. Rev. Biochem. 57:29-67.

Sternberg N, Coulby J. **1990**. Cleavage of the bacteriophage P1 packaging site (pac) is regulated by adenine methylation. Proc. Natl. Acad. Sci. USA. 87:8070-8074.

Szybalski W, Kim SC, Hasan N, Podhajska AJ. **1991**. Class-IIS restriction enzymes – a review. Gene. 100:13-26.

Trifonov EN, Sussman JL. 1980. The pitch of chromatin DNA is reflected in its nucleotide sequence. Proc. Natl. Acad. Sci. USA. 77:3816-3820.

Van der Woude MW, Braaten BA, Low DA. **1993**. Evidence for global regulatory control of pilus expression in *Escherichia coli* by Lrp and DNA methylation: model building based on analysis of *pap*. Mol. Microbiol. 6:2429-2435.

Vinella D, Jaffe A, D'Ari R, Kohiyama M, Hughes P. **1992**. Chromosome partitioning in *Escherichia coli* in the absence of Dam-directed methylation. J. Bacteriol. 174:2388-2390.

Wagner R, Messelson M. **1976**. Repair tracts in mismatched DNA heteroduplexes. Proc. Natl. Acad. Sci. USA. 73:4135-4139.

Waite-Rees PA, Keating CJ, Moran LS, Slatko BE, Hornstra LJ, Benner JS. **1991** Characterization and expression of the *Escherichia coli* Mrr restriction system. J. Bacteriol. 173:5207-5219.

Watson JD, Crick FH. **1953**. Molecular structure of nucleic acids: a structure for deoxyribose

nucleic acid. Nature. 171:737.

Watson JD, Crick FH. **1953**. Genetical implications of the structure of deoxyribonucleic acid. Nature. 171:964.

Weibauer K, Neddermann P, Hughes Melya, Jiricny J. **1993**. The repair of 5–methylcytosine deamination damage. EXS. 64:510–522.

Weissbach A. **1993**. A Chronicle of DNA methylation (1948–1975). EXS 64:1–10.

Wells RD. **1988**. Unusual DNA structures. J. Biol. Chem. 263:1095–1098.

Welsh KM, Lu AL, Clark S, Modrich P. **1987**. Isolation and characterization of the *Escherichia coli mutH* gene product. J. Biol. Chem. 262:625–629.

Yamaki H, Ohtsubo E, Nagai K, Maeda Y. **1988**. The *oriC* unwinding by *dam* methylation in *Escherichia coli*. Nucl. Acids Res. 16:5067–5073.

Yuan R, Hamilton DL. **1984**. Type I and type III restriction–modification enzymes in: DNA methylation. Eds. Razin A, Cedar H, Riggs AD. Springer–Verlag, New York. 11–37.

Zacharias W. **1993**. Methylation of cytosine influences the DNA structure. EXS. 64:27–38.

Zyskind JW, Smith DW. **1986**. The bacterial origin of replication, oriC. Cell. 46:489–490.

GENES EN REGIONES SELECCIONADAS

Genes en vacíos de GATC

Tamaño y contenido del vacío de GATC		*Rango*
Vacío de tamaño: 4839 pb		521307..526147
ybbP	putative oxidoreductase	519640..522054
rhsD	rhsD protein in rhs element	522485..526765
Vacío de tamaño: 4081 pb		3759387..3763469
yibF	putative S-transferase	3758974..3759582
rhsA	rhsA protein in rhs element	3759810..3763943
Vacío de tamaño: 3937 pb		728527..732465
ybfA	orf, hypothetical protein	728357..728563
rhsC	rhsC protein in rhs element	728806..732999
Vacío de tamaño: 3835 pb		3616646..3620482
rhsB	rhsB protein in rhs element	3616823..3621058
Vacío de tamaño: 3510 pb		2466591..2470102
b2351	putative glycan biosynthesis enzyme	2466234..2467154
b2352	putative ligase	2467151..2468482
b2353	orf, hypothetical protein	2468781..2469125
b2354	orf, hypothetical protein	2469097..2469537
yfdL	putative RNA polymerase beta	2469564..2470082
Vacío de tamaño: 3369 pb		2069370..2072740
flu	outer membrane fluffing protein, similar to adhesin	2069405..2072680

Tamaño y contenido del vacío de GATC		*Rango*
Vacío de tamaño: 3268 pb		2766105..2769374
yfjP	putative GTP–binding protein	2765725..2766594
yfjQ	orf, hypothetical protein	2766686..2767507
yfjR	orf, hypothetical protein	2767724..2768425
b2635	orf, hypothetical protein	2768310..2768702
b2636	orf, hypothetical protein	2768453..2769145
yfjT	orf, hypothetical protein	2769169..2769636
Vacío de tamaño: 2781 pb		1065698..1068480
agp	periplasmic glucose–1–phosphatase	1064808..1066049
yccJ	orf, hypothetical protein	1066087..1066314
wrbA	trp repressor binding protein	1066335..1066931
ycdF	orf, hypothetical protein	1067141..1067371
ycdG	putative transport protein	1067734..1069128
Vacío de tamaño: 2762 pb		2778338..2781101
ypjA	putative ATP–binding component of a transport system	2776167..2780876
b2648	orf, hypothetical protein	2781085..2781228
Vacío de tamaño: 2705 pb		2188008..2190714
yehB	putative outer membrane protein	2186450..2188930
yehC	putative chaperone	2188946..2189665
yehD	putative fimbrial–like protein	2189700..2190242
yehE	orf, hypothetical protein	2190535..2190816
Vacío de tamaño: 2580 pb		2650900..2653481
sseA	putative thiosulfate sulfurtransferase	2650355..2651359
sseB	enhanced serine sensitivity	2652177..2652962
pepB	putative peptidase	2653095..2654465

Tamaño y contenido del vacío de GATC		*Rango*
Vacío de tamaño: 2490 pb		1801510..1804001
b1720	orf, hypothetical protein	1801118..1801591
b1721	orf, hypothetical protein	1801602..1803017
b1722	orf, hypothetical protein	1803349..1804107
Vacío de tamaño: 2368 pb		2074302..2076671
b2001	orf, hypothetical protein	2072795..2074333
yeeS	putative DNA repair protein, RADC family	2074330..2074776
yeeT	orf, hypothetical protein	2074839..2075060
yeeU	putative structural protein	2075134..2075502
yeeV	orf, hypothetical protein	2075591..2075965
yeeW	orf, hypothetical protein	2075962..2076156
Vacío de tamaño: 2311 pb		236753..239065
dnaQ	DNA polymerase III, epsilon subunit	236067..236798
aspV	tRNA–OTHER	236931..237007
yafT	putative aminopeptidase	237335..238120
yafU	orf, hypothetical protein	238746..239084
Vacío de tamaño: 2310 pb		285763..288074
yagG	putative permease	284619..286001
yagH	putative beta–xylosidase (EC 3.2.1.37)	286013..287623
yagI	putative regulator	287628..288386
Vacío de tamaño: 2299 pb		3706928..3709228
yhjW	orf, hypothetical protein	3706413..3708137
yhjX	putative resistance protein	3708428..3709636
Vacío de tamaño: 2292 pb		864899..867192
moeA	molybdopterin biosynthesis	864352..865587
ybiK	putative asparaginase	865791..866756
b0829	putative ATP–binding component of a transport system	866776..868614

Tamaño y contenido del vacío de GATC		*Rango*
Vacío de tamaño: 2119 pb		262405..264525
b0245	orf, hypothetical protein	262552..262893
yafW	orf, hypothetical protein	262914..263231
ykfG	putative DNA repair protein	263480..263956
yafX	orf, hypothetical protein	263972..264430
Vacío de tamaño: 2105 pb		2970708..2972814
ygeD	putative resistance proteins	2970691..2971884
aas	acyl–acyl–carrier protein synthetase	2971877..2974036
Vacío de tamaño: 2059 pb		2078166..2080226
yeeA	orf, hypothetical protein	2077555..2078613
sbmC	SbmC protein	2078811..2079284
dacD	penicillin binding protein 6b	2079403..2080575
Vacío de tamaño: 2019 pb		1426786..1428806
b1371	orf, hypothetical protein	1426547..1427008
b1372	putative membrane protein	1427067..1430435

Genes en vacíos de CCWGG

Tamaño y contenido del vacío de CCWGG		*Rango*
Vacío de tamaño: 4140 pb		2753826..2757967
ssrA	tmRNA; tags incomplete translation products for degradation	2753509..2753974
intA	prophage CP4–57 integrase	2754180..2755421
yfjH	putative histone	2755665..2756621
alpA	prophage CP4–57 regulatory protein alpA	2756665..2756877
yfjI	orf, hypothetical protein	2757006..2758415

Tamaño y contenido del vacío de CCWGG		*Rango*
Vacío de tamaño: 3819 pb		2532885..2536705
ptsI	PEP–protein phosphotransferase system enzyme I	2532086..2533813
crr	PTS system, glucose–specific IIA component	2533854..2534363
pdxK	pyridoxal/pyridoxine/pyridoxamine kinase	2534406..2535257
yfeK	orf, hypothetical protein	2535362..2535736
b2420	orf, hypothetical protein	2535769..2536503
cysM	cysteine synthase B, O–acetylserine sulfhydrolase B	2536692..2537603
Vacío de tamaño: 3779 pb		3794081..3797861
rfaC	heptosyl transferase I; lipopolysaccharide core biosynthesis	3793606..3794565
rfaL	O–antigen ligase; lipopolysaccharide core biosynthesis	3794575..3795834
rfaK	lipopolysaccharide core biosynthesis	3795866..3796939
rfaZ	lipopolysaccharide core biosynthesis	3796972..3797823
Vacío de tamaño: 3192 pb		1473177..1476370
ydbD	orf, hypothetical protein	1473162..1475474
b1408	probable enzyme	1475639..1476250
b1409	putative phosphatidate cytidiltransferase	1476250..1477146
Vacío de tamaño: 3191 pb		2764516..2767708
yfjN	putative cell division protein	2763939..2765012
yfjO	orf, hypothetical protein	2765056..2765376
yfjP	putative GTP–binding protein	2765725..2766594
yfjQ	orf, hypothetical protein	2766686..2767507
Vacío de tamaño: 3128 pb		2453296..2456425
b2339	putative fimbrial–like protein	2453103..2453666
b2340	orf, hypothetical protein	2454347..2454832
b2341	putative enzyme	2455035..2457179

Tamaño y contenido del vacío de CCWGG		*Rango*
Vacío de tamaño: 3091 pb		4465982..4469074
mgtA	Mg2+ transport ATPase, P–type 1	4465199..4467895
yjgF	orf, hypothetical protein	4468101..4468526
pyrI	aspartate carbamoyltransferase, regulatory subunit	4468560..4469021
pyrB	aspartate carbamoyltransferase, catalytic subunit	4469034..4469969
Vacío de tamaño: 3017 pb		2065021..2068039
trs5_6	IS5 transposase	2064327..2065343
b1995	orf, hypothetical protein	2066630..2067049
yi22_3	IS2 hypothetical protein	2066974..2067879
yi21_3	IS2 hypothetical protein	2067837..2068247
Vacío de tamaño: 2967 pb		2278764..2281732
yejH	putative ATP–dependent helicase	2278652..2280412
rplY	50S ribosomal subunit protein L25	2280537..2280821
yejK	orf, hypothetical protein	2280960..2281967
Vacío de tamaño: 2912 pb		1389749..1392662
b1327	orf, hypothetical protein	1388957..1389889
ycjZ	putative transcriptional regulator LYSR–type	1390015..1390914
b1329	putative transport periplasmic protein	1391230..1392864
Vacío de tamaño: 2852 pb		779561..782414
ybgF	orf, hypothetical protein	778821..779612
lysT	tRNA–Lys	779777..779852
valT	tRNA–Val	779988..780063
lysW	tRNA–Lys	780066..780141
valZ	tRNA–Val	780291..780366
lysY	tRNA–Lys	780370..780445
lysZ	tRNA–Lys	780592..780667
lysQ	tRNA–Lys	780800..780875
nadA	quinolinate synthetase, A protein	781308..782351

Tamaño y contenido del vacío de CCWGG		*Rango*
Vacío de tamaño: 2815 pb		2420316..2423132
b2304	putative sugar nucleotide epimerase	2419728..2420621
yfcI	orf, hypothetical protein	2420669..2421559
hisP	ATP-binding component of histidine transport	2421756..2422529
hisM	histidine transport, membrane protein M	2422537..2423253
Vacío de tamaño: 2808 pb		3893174..3895983
yieG	putative membrane / transport protein	3892901..3894238
yieH	putative phosphatase	3894403..3895068
yieI	orf, hypothetical protein	3895135..3895602
yieJ	orf, hypothetical protein	3895651..3896238
Vacío de tamaño: 2780 pb		1103218..1105999
csgB	minor curlin subunit precursor, similar ro CsgA	1103174..1103629
csgA	curlin major subunit, coiled surface structures; cryptic	1103670..1104125
csgC	putative curli production protein	1104184..1104516
b1044	orf, hypothetical protein	1104637..1104948
b1045	putative polyprotein	1105043..1105576
ymdC	putative synthase	1105518..1106999
Vacío de tamaño: 2764 pb		2780830..2783595
ypjA	putative ATP-binding component of a transport system	2776167..2780876
b2648	orf, hypothetical protein	2781085..2781228
b2649	orf, hypothetical protein	2781658..2782449
b2650	orf, hypothetical protein	2782549..2783031
b2651	orf, hypothetical protein	2783241..2783372
Vacío de tamaño: 2641 pb		2525224..2527866
yfeH	putative cytochrome oxidase	2524966..2525964
lig	DNA ligase	2526181..2528196

Tamaño y contenido del vacío de CCWGG		*Rango*
Vacío de tamaño: 2585 pb		2989047..2991633
ygeG	orf, hypothetical protein	2989290..2989781
ygeH	putative invasion protein	2990116..2991492
Vacío de tamaño: 2570 pb		3801091..3803662
rfaB	UDP-gal(glucosyl)lipopolysacch galactosyltransferase	3800685..3801794
rfaS	lipopolysaccharide core biosynthesis	3801808..3802743
rfaP	lipopolysaccharide core biosynthesis	3802780..3803577
rfaG	glucosyltransferase I; lipopolysaccharide core biosynthesis	3803570..3804694
Vacío de tamaño: 2564 pb		3125944..3128509
glcD	glycolate oxidase subunit D	3124537..3126036
glcC	transcriptional activator for glc operon	3126287..3127051
b2981	orf, hypothetical protein	3127058..3128230
trs5_9	IS5 transposase	3128193..3129209
Vacío de tamaño: 2562 pb		1579707..1582270
b1498	putative sulfatase	1578866..1580581
b1499	putative ARAC-type regulatory protein	1580950..1581711
b1500	orf, hypothetical protein	1581786..1581983
b1501	putative oxidoreductase, major subunit	1582231..1584510
Vacío de tamaño: 2559 pb		1814739..1817299
ydjC	orf, hypothetical protein	1814410..1815159
celF	phospho-beta-glucosidase; cryptic	1815172..1816524
celD	negative transcriptional regulator of cel operon	1816629..1817471
Vacío de tamaño: 2512 pb		728864..731377
rhsC	rhsC protein in rhs element	728806..732999

Tamaño y contenido del vacío de CCWGG		*Rango*
Vacío de tamaño: 2508 pb		3687815..3690324
yhjM	putative endoglucanase	3686784..3687890
yhjN	orf, hypothetical protein	3687897..3690236
yhjO	putative cellulose synthase	3690247..3692913
Vacío de tamaño: 2506 pb		471550..474057
mdlB	putative ATP-binding component of a transport system	469860..471641
glnK	nitrogen regulatory protein P-II 2	471822..472160
amtB	probable ammonium transporter	472190..473476
tesB	acyl-CoA thioesterase II	473525..474385
Vacío de tamaño: 2486 pb		4446162..4448649
chpS	suppressor of inhibitory function of ChpB, autoregulated	4446018..4446275
chpB	probable growth inhibitor, PemK-like, autoregulated	4446269..4446619
ppa	inorganic pyrophosphatase	4446699..4447229
ytfQ	putative LACI-type transcriptional regulator	4447539..4448495
ytfR	putative ATP-binding component of a transport system	4448633..4449886

Genes en agrupamientos de GATC

Tamaño y contenido del agrupamiento de GATC		*Rango*
Agrupamiento de tamaño: 12 sitios		3923350..3923593
Agrupamiento de tamaño: 9 sitios		201458..201701
lpxD	third step of endotoxin (lipidA) synthesis	200971..201996
Agrupamiento de tamaño: 9 sitios		337248..337491
yahF	putative oxidoreductase subunit	336002..337549

Tamaño y contenido del agrupamiento de GATC		*Rango*
Agrupamiento de tamaño: 9 sitios		2565355..2565598
eutH	ethanolamine utilization	2564901..2566127
Agrupamiento de tamaño: 9 sitios		2574771..2575014
b2463	putative multimodular enzyme	2574118..2576397
Agrupamiento de tamaño: 9 sitios		2639203..2639446
gcpE	orf, hypothetical protein	2638706..2639824
Agrupamiento de tamaño: 9 sitios		2953020..2953263
recB	DNA helicase, exonuclease V subunit, ssDNA endonuclease	2950483..2954025
Agrupamiento de tamaño: 9 sitios		3062135..3062378
ygfG	putative enzyme	3061971..3062798
Agrupamiento de tamaño: 9 sitios		3342899..3343142
rpoN	RNA polymerase, sigma(54 or 60) factor	3342358..3343791
Agrupamiento de tamaño: 9 sitios		3387759..3388002
yhcS	putative transcriptional regulator LYSR-type	3387155..3388084
Agrupamiento de tamaño: 8 sitios		60891..61134
hepA	probable ATP-dependent RNA helicase	60358..63264
Agrupamiento de tamaño: 8 sitios		211243..211486
ldcC	lysine decarboxylase 2, constitutive	209679..211820
Agrupamiento de tamaño: 8 sitios		353579..353822
prpE	putative propionyl-CoA synthetase	351930..353816
Agrupamiento de tamaño: 8 sitios		693311..693554
yleA	orf, hypothetical protein	692754..694178

Tamaño y contenido del agrupamiento de GATC		*Rango*
Agrupamiento de tamaño: 8 sitios		2162338..2162581
baeR	transcriptional response regulatory protein (sensor BaeS)	2162298..2163020
Agrupamiento de tamaño: 8 sitios		2610366..2610609
hyfR	putative 2-component regulator, interaction with sigma 54	2609941..2611932
Agrupamiento de tamaño: 8 sitios		2644184..2644427
pbpC	putative peptidoglycan enzyme	2643033..2645345
Agrupamiento de tamaño: 8 sitios		2720219..2720462
yfiQ	orf, hypothetical protein	2717973..2720633
Agrupamiento de tamaño: 8 sitios		3059996..3060239
sbm	methylmalonyl-CoA mutase (MCM)	3058870..3061014
Agrupamiento de tamaño: 8 sitios		3114954..3115197
b2973	orf, hypothetical protein	3112567..3115113
b2974	putative endoglucanase	3115101..3117128
Agrupamiento de tamaño: 8 sitios		3204970..3205213
ttdA	L-tartrate dehydratase, subunit A	3204104..3205015
ttdB	L-tartrate dehydratase, subunit B	3205012..3205617
Agrupamiento de tamaño: 8 sitios		3328425..3328668
yhbZ	putative GTP-binding factor	3328223..3329395
Agrupamiento de tamaño: 8 sitios		3406058..3406301
panF	sodium/pantothenate symporter	3405238..3406695
Agrupamiento de tamaño: 8 sitios		3755681..3755924
selB	selenocysteinyl-tRNA-specific translation factor	3755644..3757488

Tamaño y contenido del agrupamiento de GATC		*Rango*
Agrupamiento de tamaño: 8 sitios		3880442..3880685
dnaA	DNA biosynthesis; initiation of replication regulator	3879954..3881357
Agrupamiento de tamaño: 8 sitios		3881487..3881730
Agrupamiento de tamaño: 8 sitios		4132494..4132737
katG	catalase; hydroperoxidase HPI(I)	4131415..4133595
Agrupamiento de tamaño: 8 sitios		4157616..4157859
udhA	putative oxidoreductase	4156969..4158303
Agrupamiento de tamaño: 8 sitios		4300321..4300564
yjcQ	putative enzyme	4298606..4300516
Agrupamiento de tamaño: 8 sitios		4312352..4312595
phnP	phosphonate metabolism	4311922..4312680
Agrupamiento de tamaño: 8 sitios		4451257..4451500
yjfF	putative transport system permease protein	4451181..4452152
Agrupamiento de tamaño: 8 sitios		4463190..4463433
treB	PTS system enzyme II, trehalose specific	4462333..4463754
Agrupamiento de tamaño: 8 sitios		4597854..4598097
dnaC	chromosome replication; initiation and chain elongation	4597807..4598544

Genes en agrupamientos de CCWGG

Tamaño y contenido del agrupamiento de CCWGG		*Rango*
Agrupamiento de tamaño: 7 sitios		112100..112486
yacF	orf, hypothetical protein	111856..112599

Tamaño y contenido del agrupamiento de CCWGG		*Rango*
Agrupamiento de tamaño: 7 sitios		777804..778190
tolB	periplasmic protein involved in the uptake of group A colicins	776963..778255
Agrupamiento de tamaño: 7 sitios		2926999..2927385
sdaC	probable serine transporter	2926251..2927540
Agrupamiento de tamaño: 7 sitios		3007583..3007969
ygeY	putative deacetylase	3006785..3007996
Agrupamiento de tamaño: 6 sitios		224392..224778
rrsH	16S ribosomal RNA	223771..225312
Agrupamiento de tamaño: 6 sitios		456040..456426
clpP	proteolytic subunit of serine protease, heat shock protein F21.5	455901..456524
Agrupamiento de tamaño: 6 sitios		708427..708813
ybfM	orf, hypothetical protein	707557..708963
Agrupamiento de tamaño: 6 sitios		746577..746963
abrB	putative transport protein	745946..747037
Agrupamiento de tamaño: 6 sitios		1374623..1375009
b1314	putative transient receptor potential locus	1374049..1374846
ycjS	putative dehydrogenase	1374856..1375911
Agrupamiento de tamaño: 6 sitios		1716265..1716651
ydhA	orf, hypothetical protein	1716090..1716338
b1640	orf, hypothetical protein	1716517..1717626
Agrupamiento de tamaño: 6 sitios		1863732..1864118
yeaF	orf, hypothetical protein	1863750..1864496

Tamaño y contenido del agrupamiento de CCWGG		*Rango*
Agrupamiento de tamaño: 6 sitios		2030689..2031075
yedJ	orf, hypothetical protein	2030406..2031101
Agrupamiento de tamaño: 6 sitios		2237011..2237397
mglA	ATP-binding component of methyl-galactoside transport	2235789..2237309
mglB	galactose-binding transport protein; receptor for galactose taxis	2237370..2238368
Agrupamiento de tamaño: 6 sitios		2332572..2332958
b2229	orf, hypothetical protein	2332356..2333006
Agrupamiento de tamaño: 6 sitios		2368275..2368661
b2256	orf, hypothetical protein	2368038..2368928
Agrupamiento de tamaño: 6 sitios		2391198..2391584
nuoL	NADH dehydrogenase I chain L	2391225..2393066
Agrupamiento de tamaño: 6 sitios		2539173..2539559
cysW	sulfate transport system permease W protein	2538824..2539273
Agrupamiento de tamaño: 6 sitios		2603943..2604329
hyfD	hydrogenase 4 membrane subunit	2602831..2604270
hyfE	hydrogenase 4 membrane subunit	2604282..2604932
Agrupamiento de tamaño: 6 sitios		2728189..2728575
rrsG	16S ribosomal RNA	2727636..2729178
Agrupamiento de tamaño: 6 sitios		3009440..3009826
yqeA	putative kinase	3009482..3010414
Agrupamiento de tamaño: 6 sitios		3074078..3074464
yggF	orf, hypothetical protein	3073237..3074202
b2931	putative oxidoreductase	3074199..3075188

Tamaño y contenido del agrupamiento de CCWGG		*Rango*
Agrupamiento de tamaño: 6 sitios		3075902..3076288
cmtA	PTS system, mannitol–specific enzyme II component, cryptic	3075490..3076878
Agrupamiento de tamaño: 6 sitios		3223082..3223468
ebgA	evolved beta–D–galactosidase, alpha subunit; cryptic gene	3220238..3223366
ebgC	evolved beta–D–galactosidase, beta subunit; cryptic gene	3223363..3223812
Agrupamiento de tamaño: 6 sitios		3227555..3227941
ygjK	putative isomerase	3226529..3228880
Agrupamiento de tamaño: 6 sitios		3260346..3260732
tdcD	putative kinase	3260093..3261313
Agrupamiento de tamaño: 6 sitios		3425411..3425797
rrsD	16S ribosomal RNA	3424858..3426399
Agrupamiento de tamaño: 6 sitios		3475827..3476213
slyD	FKBP–type peptidyl–prolyl cis–trans isomerase (rotamase)	3475544..3476134
Agrupamiento de tamaño: 6 sitios		3696089..3696475
yhjU	orf, hypothetical protein	3695843..3697522
Agrupamiento de tamaño: 6 sitios		3844577..3844963
uhpT	hexose phosphate transport protein	3843403..3844794
uhpC	regulator of uhpT	3844932..3846254
Agrupamiento de tamaño: 6 sitios		3913335..3913721
atpC	membrane–bound ATP synthase, F1 sector, epsilon–subunit	3913181..3913600
atpD	membrane–bound ATP synthase, F1 sector, beta–subunit	3913621..3915003
Agrupamiento de tamaño: 6 sitios		3940052..3940438
rrsC	16S ribosomal RNA	3939431..3940971

Tamaño y contenido del agrupamiento de CCWGG		*Rango*
Agrupamiento de tamaño: 6 sitios		4033741..4034127
rrsA	16S ribosomal RNA	4033120..4034661
Agrupamiento de tamaño: 6 sitios		4164859..4165245
rrsB	16S ribosomal RNA	4164238..4165779
Agrupamiento de tamaño: 6 sitios		4206346..4206732
rrsE	16S ribosomal RNA	4205725..4207266
Agrupamiento de tamaño: 6 sitios		4335781..4336167
adiA	biodegradative arginine decarboxylase	4335832..4338102
Agrupamiento de tamaño: 6 sitios		4445272..4445658
ytfN	orf, hypothetical protein	4441689..4445468
ytfP	orf, hypothetical protein	4445471..4445812
yjfA	orf, hypothetical protein	4445553..4445822
Agrupamiento de tamaño: 6 sitios		4572415..4572801
yjiU	orf, hypothetical protein	4571704..4573245

PROGRAMAS

#Prog	*Página*	*Propósito*
08	P15	Para cada ventana, calcula el valor de probabilidad encontrada por la integración de una curva de distribución normal desde el valor que tiene dicha ventana hasta terminar la cola de distribución, ya sea positiva o negativa, de acuerdo a si está por arriba o por abajo de la media de los datos observados.
09	P17	Grafica una representación circular del genoma de *E. coli*, mostrando todas las ventanas con el logaritmo del inverso de su probabilidad. Esta conversión es necesaria para que los picos con probabilidad más pequeña (es decir, más significativa) resalten.
10	P24	Obtiene una lista de las posiciones donde existen sitios de metilación de ambos tipos.
11	P25	Calcula el espacio que existe entre cada sitio de metilación y ordena dichos espacios (vacíos) de mayor a menor.
12	P26	Con un tamaño igual a la frecuencia observada de cada tipo de sitio de metilación, busca en el genoma los 100 fragmentos con mayor número de sitios (agrupamientos).
13	P28	Calcula la frecuencia de ambos tipos de metilación en tres regiones del genoma: traducido, solamente transcrito e intergénico.
14	P29	Obtiene el porcentaje de G–C en cada una de las tres regiones mencionadas en el programa anterior.
15	P30	Toma todas las posiciones metilables, es decir, dos por sitio y cuenta cuantas veces corresponden a la 1ra, 2da o 3ra base del codón, así como en regiones no traducidas y regiones no transcritas.

#Prog	*Página*	*Propósito*
16	P33	Considera todos los genes de *E. coli* y para cada uno de ellos, cuenta los sitios de metilación Dcm que contiene. Adicionalmente calcula el promedio por base de los sitios Dam que pueden actuar sobre el gen o la cantidad de bases que simplemente no puedan ser protegidas por Dam, en caso de que haya.
17	P34	A partir de una lista de genes esenciales o no esenciales, extrae sus "b–numbers" y con estos extrae su información completa del archivo de información sobre la metilación de cada gen generada con el programa 17.
18	P35	Toma las listas de vacíos y agrupamientos, escogidos como candidatos a ser analizados, y encuentra todos los genes que coincidan aunque sea parcialmente con la región en cuestión.

Programa 01 (lab/pela)

```
#!/usr/bin/perl

# Opening files and coments on usage

open(IN,"@ARGV[0]") || die "usage: pela INPUT OUTPUT (INPUT must exist)\n";
if (-e @ARGV[1]) {
  print "\"@ARGV[1]\" already exists, are you sure? "; chomp($choice=<STDIN>);
  exit(0) unless $choice =~ /^y/i;
}
open(OUT,">@ARGV[1]") || die "usage: pela INPUT OUTPUT (coudn't create \"@ARGV[1]\")\n";
open(OUT_LINE,">@ARGV[1]_line") || die "coudn't create \"@ARGV[1]_line\".\n";


# Get only lines from ORIGIN on, and send with or without stuff to files

$first_line = <IN>;

foreach (<IN>) {
  if ($found ==1) {
    print OUT;
    s/[\s+\d+\/]//g;
    print OUT_LINE;
    $length += length;
  }
  $found = 1 if /^ORIGIN/;
}

# Check and compare lengths

($length_db,$check) = (split(/\s+/,$first_line))[2,3];
die "An ugly death. Length doesn't match!\n" if ($check ne "bp");
print "The original DB had $length_db $check, your line has $length.\n";

close(IN);
close(OUT);
close(OUT_LINE);
```

Programa 02 (lab/x)

```
#!/usr/bin/perl

## Start all over again, needs to be in window size, careful for (aa's in aaaa)
## Forgets extra bases, but tells you how many.

die "usage: x FILE\n" unless –e ($file = @ARGV[0]);

open(OUT,">>ecoli.log") || die "coudn't create log file!\n";
$db = <>;

# IUPAC formats, in a hash.
%iupac =("n","[atgc]","b","[cgt]","d","[agt]","h","[act]","v","[acg]",
         "k","[gt]","y","[ct]","s","[cg]","w","[at]","r","[ag]","m","[ac]");

print "What string do you want to search for? "; chomp($string = <STDIN>);
die "You have to give me a string!\n" if $string eq "";

# Just a check in case things like aa are asked.
$one = $two = $string;
$one =~ s/^.//;
$two =~ s/.$//;
$check = $one . $two;

if ($check =~ /$string/) {
        print "Cound would be lost on string intersect, do you want to continue? ";
        chomp($answer = <STDIN>);
        die "\n" if $answer =~ /^n/i;
        }

# Process string
     $string2 = $string;
        $string2 =~ s/\[.+?\]/x/g; $len_str = length$string2;
        $string =~ tr/A–Z/a–z/;
     $string =~ s/$_/$iupac{$_}/g foreach (keys%iupac);

# Process db
        $len_db = length$db;
        $db .= substr($db,0,$len_str –1);

print "What window size do you want? "; chomp($win_size = <STDIN>);
        $win_size = $len_db if $win_size == "";
     $win_size = $len_db if $win_size > $len_db;
     $real_size = $win_size + $len_str –1;

# Check for extra bases
        &ventana;
        $extra = $len_db % $win_size;
        $win_amount = int($len_db / $win_size);

$old = select(OUT);
$= = $win_amount + 8;
select($old);

$sd_sum = $total = $highest = $cycles = 0;
$lowest = $win_size;
```

```
for ($start = 0; length($window = substr($db,$start,$real_size)) == $real_size; $start += $win_size) {
        $string3 = $string; $count = 0;
        $total += $count = $window =~ s/$string//g;
        $highest = $count if $count > $highest;
        $lowest = $count if $count < $lowest;
        $sum[$cycles] = $count; $cycles++;
        write OUT;
        }

$mean = $total / $cycles;
($sd_sum += ($_ - $mean) ** 2) foreach @sum;
$stdev = ($sd_sum / $cycles) ** 0.5;

print "You lost $extra bases. The mean was $mean and the SD $stdev.
The total hits were $total.\n";
print "The lowest was $lowest, and the highest $highest.\n";

format OUT =
 @#####    @#########     @######## - @########
$cycles, $count, $start + 1, $start + $win_size
.

format OUT_TOP =

-------------------------------------------------
 You searched for:  ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$string3
~~              ^<<<<<<<<<<<<<<<<<<<<<<<<<<<<
$string3

 Window size: @#######            Total bases: @######
$win_size,$len_db

 Window    Amount found      Window position
--------   ------------   ----------------------
.

sub ventana {
# Calculate safe window size
$n = $len_db;
$size = $win_size;
$extra = $n % $size;
$allow = $size / 10;

if ($extra > $allow) {
        $Msize = $Lsize = $size;
        while (1 == 1) {
                $Msize++; $Lsize--;
                $extraM = $n % $Msize; $extraL = $n % $Lsize;
                if ($extraM < $allow) {$ok1 = $Msize;$Msize--}
                if ($extraL < $allow) {$ok2 = $Lsize;$Lsize++}
                last if ($ok1 and $ok2);
                }
        die "Ouch! With $size you lose $extra, but don't fear!
with $ok1 you lose $extraM and with $ok2 you lose $extraL.\n";
}}
```

Programa 03 (lab/get_distribution)

```
#!/usr/bin/perl

## Need to compare values from each window in each case and assign a value.


@number = ("02509","04999","09998","19996","39993");

foreach $number (@number) {

$a_results = "TRI_RESULTS_$number";          # Name of results file
$c_results = "TETRA_RESULTS_$number";        # Name of results file
$path = "Results/$number/";                  # Set path to find files
$output = "distrib_markovian_$number";       # Name of output file
$output2 = "distrib_real_$number";           # Name of output file
$a_ecoli = "A_ecoli_$number";
$c_ecoli = "C_ecoli_$number";

open(AR,"<$path$a_results") || die "Coudn't open file!\n";
open(CR,"<$path$c_results") || die "Coudn't open file!\n";

$meanA = $sdA = $meanC = $sdC = $cA = $cC = 0;

while (<AR>) {
        @data = split;
        if (/^gatc/) {$meanA += $data[1]; $sdA += $data[2]; $cA++}
        }
while (<CR>) {
        @data = split;
     if (/^cc/) {$meanC += $data[1]; $sdC += $data[2]; $cC++}
        }

$meanA /= $cA;
$meanC /= $cC;
$sdA /= $cA;
$sdC /= $cC;

open(OUT,">$path$output") || die "Coudn't create file!\n";
printf OUT "  String\t Mean\t   SD
----------    -------- --------
cc[at]gg\t%3.3f\t %2.3f
gatc    \t%3.3f\t %2.3f
",$meanC,$sdC,$meanA,$sdA;


open(EAR,"<$path$a_ecoli") || die "Coudn't open file!\n";
open(ECR,"<$path$c_ecoli") || die "Coudn't open file!\n";

$sd_sum = $count = $total = 0; undef @sum; undef @data;
while (<EAR>) {
        @data = split;
        next unless $data[0] =~ /\d+/;
        $total += $sum[$count] = $data[1];
        $count++;
        }
$meanA = $total / $count;
```

```
($sd_sum += ($meanA - $_) ** 2) foreach @sum;
$sdA = sqrt($sd_sum / $count);

$sd_sum = $count = $total = 0; undef @sum; undef @data;
while (<ECR>) {
        @data = split;
        next unless $data[0] =~ /\d+/;
        $total += $sum[$count] = $data[1];
        $count++;
        }
$meanC = $total / $count;
($sd_sum += ($meanC - $_) ** 2) foreach @sum;
$sdC = sqrt($sd_sum / $count);

open(OUT,">$path$output2") || die "Coudn't create file!\n";
printf OUT "  String\t Mean\t   SD
----------    -------- --------
cc[at]gg\t%3.3f\t %2.3f
gatc    \t%3.3f\t %2.3f
",$meanC,$sdC,$meanA,$sdA;

}
```

Programa 04 (lab/pentanuc)

```
#!/usr/bin/perl

## Generates table of penta, tetra, tri, di and nucleotide usage for a given genome.

$ARGV[0] = "DB/ecoli_line";                # Remove later
die "usage: dinuc FILE\n" unless -e ($file = @ARGV[0]);

$db = <>;
$nuc1 = substr($db,0,1);
$nuc2 = substr($db,1,1);
$nuc3 = substr($db,2,1);
$nuc4 = substr($db,3,1);
$dinuc1 = substr($db,0,2);
$dinuc2 = substr($db,1,2);
$dinuc3 = substr($db,2,2);
$tri1 = substr($db,0,3);
$tri2 = substr($db,1,3);
$tetra1 = substr($db,0,4);
$db .= $tetra1;
@base = ("a","c","g","t");

foreach $first (@base) {
        $nuc{$first} = &find($first);
        foreach $second (@base) {
                $dinuc = $first . $second;
                $dinuc{$dinuc} = &find($dinuc);
                foreach $third (@base) {
                        $tri = $dinuc . $third;
                        $tri{$tri} = &find($tri);
                        foreach $fourth (@base) {
                                $tetra = $tri . $fourth;
                                $tetra{$tetra} = &find($tetra);
                                foreach $fifth (@base) {
                                        $penta = $tetra . $fifth;
                                        $penta{$penta} = &find($penta);
                                }
                        }
                }
        }
}

# Remove counts for nucleotides added to dbase
$nuc{$nuc1}--;$nuc{$nuc2}--;$nuc{$nuc3}--;$nuc{$nuc4}--;
$dinuc{$dinuc1}--;$dinuc{$dinuc2}--;$dinuc{$dinuc3}--;
$tri{$tri1}--;$tri{$tri2}--;
$tetra{$tetra1}--;

open(NUC, ">nuc-coli")                  || die "coudn't create!\n";
open(DINUC, ">dinuc-coli")              || die "coudn't create!\n";
open(TRI, ">trinuc-coli")               || die "coudn't create!\n";
open(TETRA, ">tetranuc-coli")           || die "coudn't create!\n";
open(PENTA, ">pentanuc-coli")           || die "coudn't create!\n";

# Prints pentanucleotide usage
foreach (sort keys %penta) {
```

```
        printf PENTA "$_\t%7d\n",$penta{$_};
    }

# Prints tetranucleotide usage
foreach (sort keys %tetra) {
        printf TETRA "$_\t%7d\n",$tetra{$_};
    }

# Prints trinucleotide usage
foreach (sort keys %tri) {
        printf TRI "$_\t%7d\n",$tri{$_};
    }

# Prints dinucleotide usage
foreach (sort keys %dinuc) {
        printf DINUC "$_\t%7d\n",$dinuc{$_};
    }

# Prints nucleotide usage
foreach (sort keys %nuc) {
        printf NUC "$_\t%7d\n",$nuc{$_};
    }

close(NUC);
close(DINUC);
close(TRI);
close(TETRA);
close(PENTA);


sub find {
        local($count,$where) = 0;
        $count++ while ($where = (index($db,$_[0],$where))+1);
        return $count;
        }
```

Programa 05 (lab/histo)

```
#!/usr/bin/perl

## Get 2nd column and make a histogram out of frequencies.

$file = $ARGV[0];
while (<>) {
        next unless /^\s+\d+/;
        @data = split;
        $data{$data[1]}++;
}

@keys = sort {$a <=> $b} keys %data;
$last = pop@keys;
$first = shift@keys;

print "$file\n\n";
for ($_ = $first; $_ <= $last; $_++) {
        write;
}


format STDOUT =
@#### @####
$_,$data{$_}
.
```

Programa 06 (lab/oder_lines)

```
#!/usr/bin/perl

## Have to sort out the lines from *_ecoli_# to *_ecoli_sorted

@sizes = ("02509","04999","09998","19996","39993");


foreach $letter ("A","C") {

foreach $size (@sizes) {
# Get means
        open(DISTRIB,"<Results/$size/distrib_real_$size") || die "No distrib file!\n";
                while (<DISTRIB>) {
                        ($C = (split)[1]) if (/^cc/);
                        ($A = (split)[1]) if (/^gatc/);
                }
open(IN,"<Results/$size/$letter\_ecoli_$size") || die "Can't open file!\n";
open(OUT,">Results/$size/$letter\_ecoli_sorted") || die "Can't create file!\n";

undef @data; undef @info;

while (<IN>) {
        unless ((split)[0] =~ /^\d+/) {print OUT; next}
        @data = split;
        push(@info,"$data[1]\t$data[0]\t$data[1]\t$data[2] $data[3] $data[4]\n");
}
undef @data;
foreach (sort wierd @info) {
        @data = split;
        print OUT "  $data[1]\t\t$data[2]\t\t$data[3] $data[4] $data[5]\n";
}

}}



sub wierd {
        if ($a < ${$letter} && $b < ${$letter}) {
                $a <=> $b;
        } else {
                $b <=> $a;
        }
}
```

Programa 07 (lab/afinar_all)

```
#!/usr/bin/perl

## Use DB/positions and ALL windows to get highest count

@sizes = ("02509","04999","09998","19996","39993");

foreach $letter ("A","C") {

foreach $size (@sizes) {

# Get means
        open(DISTRIB,"<Results/$size/distrib_real_$size") || die "No distrib file!\n";
                while (<DISTRIB>) {
                        ($C = (split)[1]) if (/^cc/);
                        ($A = (split)[1]) if (/^gatc/);
                }

# Open the files
        open(COLI_IN,"<Results/$size/$letter\_ecoli_sorted") || die "No original file!\n";
        open(COLI_OUT,">Results/$size/2$letter\_ecoli_$size") || die "Can't create file!\n";

# Get 0s&1s vector for methylation positions, reget because I over-right
        undef @positions; $positions[4639220] = 0;
        open(POS,"<DB/positions_$letter") || die "No positions file!\n";
        while (<POS>) {$positions[$_] = 1}
        @end = @positions[-20000..-1];
        @positions = (@positions,@positions[0..19999],@end);

# Cycle for highests
        while (<COLI_IN>) {
                undef @data;
                @data = split;
                unless ($data[0] =~ /^\d+/) {print COLI_OUT $_; next} # Leave the header alone
                if ($data[1] > ${$letter}) {
                        $new = &hits_high;
                        print COLI_OUT "   $data[0]\t\t$new\t\t$data[2] $data[3] $data[4]\t\t$pos\n";
                } else {last}
        }

# Reopen the input file
        open(COLI_IN,"<Results/$size/$letter\_ecoli_sorted") || die "No original file!\n";

# Get 0s&1s vector for methylation positions, reget because I over-right
        undef @positions; $positions[4639220] = 0;
        open(POS,"<DB/positions_$letter") || die "No positions file!\n";
        while (<POS>) {$positions[$_] = 1}
        @end = @positions[-20000..-1];
        @positions = (@positions,@positions[0..19999],@end);

# Cycle for lowests
        while (<COLI_IN>) {
                undef @data;
                @data = split;
                next unless ($data[0] =~ /^\d+/); # Header is already written
                if ($data[1] <= ${$letter}) {
```

```
                        $new = &hits_low;
                        print COLI_OUT "   $data[0]\t\t$new\t\t$data[2] $data[3] $data[4]\t\t$pos\n";
                }
        }
}}

sub hits_high {
        undef @window; undef @rest;
        $start = int($data[2]-$size/2);
        $end = $start+2*$size-1;
        @window = @positions[$start..$start+$size-1];
        @rest = @positions[$start+$size..$end];
        $hits = $c = 0;
        $highest = $data[1]; # or change to $hits after 1st foreach cycle
        $pos = $data[2]-1;
        foreach (@window) {$hits += $_}
        foreach (0..$size-1) {
                $hits += $rest[$_];
                $hits -= $window[$_]; $c++;
                if ($hits > $highest) {$highest = $hits; $pos = $start + $c}
        }
                foreach($pos..$pos+$size-1) {
                        $positions[$_] = 0;
                }
        return $highest;
}

sub hits_low {
        undef @window; undef @rest;
        $start = int($data[2]-$size/2);
        $end = $start+2*$size-1;
        @window = @positions[$start..$start+$size-1];
        @rest = @positions[$start+$size..$end];
        $hits = $c = 0;
        $lowest = $data[1]; # or change to $hits after 1st foreach cycle
        $pos = $data[2]-1;
        foreach (@window) {$hits += $_}
        foreach (0..$size-1) {
                $hits += $rest[$_];
                $hits -= $window[$_]; $c++;
                if ($hits < $lowest) {$lowest = $hits; $pos = $start + $c}
        }
                foreach($pos..$pos+$size-1) {
                        $positions[$_] = 1;
                }
        return $lowest;
}
```

Programa 08 (lab/get_prob_afinado)

```
#!/usr/bin/perl

## Need to compare values from each window in each case and assign a value.

$pi = atan2(1,1) *4;
$step = 0.01;                                   # interval value for integration
@number = ("02509","04999","09998","19996","39993");


foreach $number (@number) {

$a_ecoli = "2A_ecoli_$number";                  # Ecoli data file
$c_ecoli = "2C_ecoli_$number";                  # Ecoli data file
$path = "Results/$number/";                     # Set path to find files
$dis_mark = "distrib_markovian_$number";        # Name of output file
$dis_ecoli = "distrib_real_$number";
$a_out = "2A_prob_$number";
$c_out = "2C_prob_$number";

$meanA = $sdA = $meanC = $sdC = 0;

open(DATA,"<$path$dis_mark") || die "Coudn't open file!\n";
open(DATA2,"<$path$dis_ecoli") || die "Coudn't open file!\n";

undef@data;
while (<DATA>) {
        @data = split;
        if (/^cc/) {$meanC = $data[1]; $sdC = $data[2]}
        if (/^gatc/) {$meanA = $data[1]; $sdA = $data[2]}
}
undef@data;
while (<DATA2>) {
        @data = split;
        if (/^cc/) {$meanC2 = $data[1]; $sdC2 = $data[2]}
        if (/^gatc/) {$meanA2 = $data[1]; $sdA2 = $data[2]}
}

# Now I need to get the ecoli files and start making the integrations
open(AIN,"<$path$a_ecoli") || die "Coudn't open file!\n";
open(CIN,"<$path$c_ecoli") || die "Coudn't open file!\n";
open(AOUT,">$path$a_out") || die "Coudn't create file!\n";
open(COUT,">$path$c_out") || die "Coudn't create file!\n";
undef@data;

print AOUT
"  Window     Amount found       Markov distrib            E.coli distrib
----------  --------------       ----------------  ----------------\n";

print COUT
"  Window     Amount found       Markov distrib            E.coli distrib
----------  --------------       ----------------  ----------------\n";

$media = $meanA;                $media2 = $meanA2;
$stdev = $sdA;                  $stdev2 = $sdA2;
$end = $media + (15 * $stdev);  $end2 = $media2 + (15 *$stdev2);
```

```
while (<AIN>) {
        @data = split;
        next unless ($data[0] =~ /\d+/);
        $start = $start2 = $data[1];
        if ($start <= $media) {$start = 2 * $media - $start}
        $prob = &method_a;
        if ($start2 <= $media2) {$start2 = 2 * $media2 - $start2}
        $prob2 = &method_a2;
        printf AOUT " %4d\t\t %4d\t\t\t%1.20f\t%1.20f\n",$data[0],$data[1],$prob,$prob2;
}

$media = $meanC;                $media2 = $meanC2;
$stdev = $sdC;                  $stdev2 = $sdC2;
$end = $media + (15 * $stdev);  $end2 = $media2 + (15 *$stdev2);
while (<CIN>) {
        @data = split;
        next unless ($data[0] =~ /\d+/);
        $start = $start2 = $data[1];
        if ($start <= $media) {$start = 2 * $media - $start}
        $prob = &method_a;
        if ($start2 <= $media2) {$start2 = 2 * $media2 - $start2}
        $prob2 = &method_a2;
        printf COUT " %4d\t\t %4d\t\t\t%1.20f\t%1.20f\n",$data[0],$data[1],$prob,$prob2;
}
}


sub function {$p = 1/ (sqrt(2*$pi) * $stdev * exp(0.5 * ((($_[0] - $media) / $stdev) ** 2)))}

sub function2 {$p = 1/ (sqrt(2*$pi) * $stdev2 * exp(0.5 * ((($_[0] - $media2) / $stdev2) ** 2)))}

sub method_a { .        # trapezoidal para cola derecha
  $cycles = $sum = 0;
  for ($s = $start + $step; $s <= ($end - $step); $s += $step) {
        $sum += &function($s);
        $cycles++;
  }
  $int = (($end - $start) * (&function($start) + &function($end) + (2 * $sum))) / (($cycles + 1) * 2);
}

sub method_a2 {         # trapezoidal para cola derecha
  $cycles = $sum = 0;
  for ($s = $start2 + $step; $s <= ($end2 - $step); $s += $step) {
        $sum += &function2($s);
        $cycles++;
  }
  $int = (($end2 - $start2) * (&function2($start2) + &function2($end2) + (2 * $sum))) / (($cycles + 1) * 2);
}
```

Programa 09 (lab/2circle)

```
#!/usr/bin/perl

## Put sorted data into log scale circle charts... (Scales OK)
##  ** FIXED COMBINED CIRCLES **

# GD library
use GD;

# Define circle and conversion of data into radians, and constants
        $pi = atan2(1,1) *4;
        $imag_size = 1200;                              # Image size
        $r = 260;                                       # Radius
        $rmax = 245;                                    # Radius of data
        $h = $k = $imag_size / 2;                       # x and y offset
        $max = 4639221;                                 # Last base
        $factor = $pi * 2/$max;
        $origin = (3923372 + 3923603) /2 * $factor;
        $offset = $origin - 3/2 * $pi;

# Set up cycles, for each size and each methylated nuc.
@sizes = ("02509","04999","09998","19996","39993");

foreach $size (reverse@sizes) {
        open(A,"<Results/$size/2A_prob_$size") || die "Coudn't open file!\n";
        open(C,"<Results/$size/2C_prob_$size") || die "Coudn't open file!\n";
        open(DIST,"<Results/$size/distrib_real_$size") || die "Coudn't open file!\n";
        while (<DIST>) {
                if (/^cc/) {($mean_C) = (split)[1]};
                if (/^ga/) {($mean_A) = (split)[1]};
        }

        foreach $key ("A","C","both") { # Start second cycle, for each case
                undef%extras; undef%original_extras;
                unless ($key eq "both") {
                  open(OUT,">Results/$key\_$size\_all.png") || die "Can't open... arghh!\n";
                } else {
                  open(OUT2,">Results/combined_$size\_all.png") || die "Can't open... arghh!\n";
        }

# Create image and define colors
        $image = new GD::Image($imag_size,$imag_size+100);
        &define_colors;

        $brush3 = new GD::Image(2,2);                   # Slightly larger brush, for labels
        $brush3 -> colorAllocate(255,0,0);              # of extras
        $image -> setBrush($brush3);

# Give max value for each window
if ($key eq "C") {
        $base = 27;
        if ($size eq "02509") {$value_times = $base}
                elsif ($size eq "04999") {$value_times = $rmax/(log(1/2) + $rmax/$base)}
                elsif ($size eq "09998") {$value_times = $rmax/(log(1/4) + $rmax/$base)}
                elsif ($size eq "19996") {$value_times = $rmax/(log(1/8) + $rmax/$base)}
                elsif ($size eq "39993") {$value_times = $rmax/(log(1/16) + $rmax/$base)}
```

```
        $x1 = 20; $y1 = 1200;                                   # get upper left corner
        $x2 = $y2 = 20;                                         # box size
        $image -> dashedLine($x1,$y1+$y2/2,$x1+$x2,$y1+$y2/2,gdBrushed);
                $image -> string(gdGiantFont,$x1+30,$y1+2,"Out of scale values",$black);
                $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$green);
                $image -> string(gdGiantFont,$x1+30,$y1+2,"Above average cytosine
        methylation",$black);
                $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$cyan);
                $image -> string(gdGiantFont,$x1+30,$y1+2,"Below average cytosine
        methylation",$black);
                $y1 +=30;

} elsif ($key eq "A") {
        $base = 22;
        if ($size eq "02509") {$value_times = $base}
                elsif ($size eq "04999") {$value_times = $rmax/(log(1/2) + $rmax/$base)}
                elsif ($size eq "09998") {$value_times = $rmax/(log(1/4) + $rmax/$base)}
                elsif ($size eq "19996") {$value_times = $rmax/(log(1/8) + $rmax/$base)}
                elsif ($size eq "39993") {$value_times = $rmax/(log(1/16) + $rmax/$base)}

        $x1 = 20; $y1 = 1200;                                   # get upper left corner
        $x2 = $y2 = 20;                                         # box size
        $image -> dashedLine($x1,$y1+$y2/2,$x1+$x2,$y1+$y2/2,gdBrushed);
                $image -> string(gdGiantFont,$x1+30,$y1+2,"Out of scale values",$black);
                $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$green);
                $image -> string(gdGiantFont,$x1+30,$y1+2,"Above average adenine
        methylation",$black);
                $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$cyan);
                $image -> string(gdGiantFont,$x1+30,$y1+2,"Below average adenine
        methylation",$black);
                $y1 +=30;

} elsif ($key eq "both") {
        $base = 17;
        if ($size eq "02509") {$value_times = $base}
                elsif ($size eq "04999") {$value_times = $rmax/(log(1/2) + $rmax/$base)}
                elsif ($size eq "09998") {$value_times = $rmax/(log(1/4) + $rmax/$base)}
                elsif ($size eq "19996") {$value_times = $rmax/(log(1/8) + $rmax/$base)}
                elsif ($size eq "39993") {$value_times = $rmax/(log(1/16) + $rmax/$base)}

# How about labels?
        $x1 = 20; $y1 = 1130;                                   # get upper left corner
        $x2 = $y2 = 20;                                         # box size
        $image -> dashedLine($x1,$y1+$y2/2,$x1+$x2,$y1+$y2/2,gdBrushed);
        $image -> string(gdGiantFont,$x1+30,$y1+2,"Out of scale values",$black); $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$darkgreen);
        $image -> string(gdGiantFont,$x1+30,$y1+2,"Below average A, above average  C",$black); $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$green);
        $image -> string(gdGiantFont,$x1+30,$y1+2,"Below average C, above average  A",$black); $y1 +=30;
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$blue);
        $image -> string(gdGiantFont,$x1+30,$y1+2,"Both methylation types below       average",$black); $y1
+=30;
```

```
        $image -> filledRectangle($x1,$y1,$x1+$x2,$y1+$y2,$cyan);
        $image -> string(gdGiantFont,$x1+30,$y1+2,"Both methylation types above      average",$black); $y1
+=30;


# Use this cycle to graph combined data, FIXED FOR COMBINED
        foreach (sort {$a <=> $b} keys %A) {

                $value = log(1/($A{$_} * $C{$_})) * $value_times;
                $pos = $_; $pos2 = $pos{$_};

                if (($A_sign{$_} eq "N") and ($C_sign{$_} eq "N")) {
                        $color = $blue;

                } elsif (($A_sign{$_} eq "P") and ($C_sign{$_} eq "P")) {
                        $color = $cyan;

                } elsif (($A_sign{$_} eq "N") and ($C_sign{$_} eq "P")) {
                        $value *= -1; $color = $darkgreen;

                } elsif (($A_sign{$_} eq "P") and ($C_sign{$_} eq "N")) {
                        $value *= -1; $color = $green;

                } else {die "a wierd death!\n"}

                if (abs($value) > $rmax) {
                        $original_extras{$pos} = abs($value);
                        $value = $value/abs($value) * $rmax;
                        $extras{$pos} = $value;
                        }
                undef @data; $c = 0;
                for ($s = $pos; $s < $pos2; $s += 0.01) {
                        $data[$c] = $s;
                        $c++;
                }
                $data[$c] = $pos2;
                $image -> filledPolygon(&make_poly(@data),$color);
        }
}


# Get data to print
$count = 0; undef %{$key}; undef %pos; undef %{$key.'_sign'}; # FIXED FOR COMBINED
        while (<$key>) {
                @columns = split;
                next unless $columns[0] =~ /\d+/;
                ($pos,$value) = (($columns[0]-1) * $size, log(1/($columns[3] * 2)));
                $pos = $pos * $factor - $offset;
                $pos2 = $pos + $size * $factor;

                        ${$key}{$pos} = ($columns[3] *2);       # For combined data
                        $pos{$pos} = $pos2;                     # FIXED FOR COMBINED

                $value *= $value_times;
                if ($columns[1] < ${'mean_'.$key}) {
                        $value *= -1;
```

```
                ${$key.'_sign'}{$pos} =  "N";              # FIXED FOR COMBINED
                $color = $cyan;
                } else {
                $color = $green;
                ${$key.'_sign'}{$pos} = "P";               # FIXED FOR COMBINED
        }
        if (abs($value) > $rmax) {
                $original_extras{$pos} = abs($value);
                $value = $value/abs($value) * $rmax;
                $extras{$pos} = $value;
        }
        undef @data; $c = 0;
        for ($s = $pos; $s < $pos2; $s += 0.01) {
                $data[$c] = $s;
                $c++;
        }
        $data[$c] = $pos2;
        $image -> filledPolygon(&make_poly(@data),$color);
}

# Need to draw genes in outer circle
        open(GENES,"<DB/all-genes-ecoli") || die "No gene DB!\n";
        $r2 = 530;
        $value = 10;
        while (<GENES>) {
                ($range,$dir) = (split)[4,5];
                ($pos,$pos2) = split(/\.\./,$range);
                $pos = $pos * $factor - $offset;
                $pos2 = $pos + $size * $factor;
                $pos = ($pos + $pos2) /2;
                if ($dir eq "F") {
                        &pos_by_angle($pos,$r2); $x1 = $x; $y1 = $y;
                        &pos_by_angle($pos,$r2 + $value); $x2 = $x; $y2 = $y;
                        $image -> line($x1,$y1,$x2,$y2,$orange);
                } else {
                        &pos_by_angle($pos,$r2); $x1 = $x; $y1 = $y;
                        &pos_by_angle($pos,$r2 - $value); $x2 = $x; $y2 = $y;
                        $image -> line($x1,$y1,$x2,$y2,$yellow);
                }
        }

# Set brush
        $brush = new GD::Image(3,3);
        $brush -> colorAllocate(130,130,130);
        $bgray = $brush -> colorAllocate(170,170,190);
        $image -> setBrush($brush);
        $r3 = $r2 + 35;

# Draw half-circle arrows
        $ang = 4; $ang_rad = $ang * 2 * $pi / 360;
        $arrow = 12 * $pi /360;
        $image -> arc($h,$k,2 * $r3,2 * $r3,270+$ang,90-$ang,gdBrushed);
        $image -> arc($h,$k,2 * $r3,2 * $r3,90+$ang,270-$ang,gdBrushed);

# Draw arrow heads:
        &pos_by_angle(1/2*$pi - $ang_rad,$r3); $xpr = $x; $ypr = $y;
```

```
        &pos_by_angle(1/2*$pi + $ang_rad,$r3); $xpl = $x; $ypl = $y;

        &pos_by_angle(1/2*$pi - $arrow,$r3+5); $x1r = $x; $y1r = $y;
        &pos_by_angle(1/2*$pi + $arrow,$r3+5); $x1l = $x; $y1l = $y;
        &pos_by_angle(1/2*$pi - $arrow,$r3-5); $x2r = $x; $y2r = $y;
        &pos_by_angle(1/2*$pi + $arrow,$r3-5); $x2l = $x; $y2l = $y;

        $image -> line($xpl,$ypl,$x1l,$y1l,gdBrushed);
        $image -> line($xpl,$ypl,$x2l,$y2l,gdBrushed);
        $image -> line($xpr,$ypr,$x1r,$y1r,gdBrushed);
        $image -> line($xpr,$ypr,$x2r,$y2r,gdBrushed);

# Draw circle with defined data, quadrants, and origin/end of rep
        $image -> arc($h,$k,2 * $r,2 * $r,0,360,$line_color);
        $image -> arc($h,$k,2 * $r2,2 * $r2,0,360,$line_color);
        $image -> line($h,$k - $r2,$h,$k + $r2,$line_color);
        $image -> line($h - $r2,$k,$h + $r2,$k,$line_color);

# Draw and annotate info (TEXT)
                &pos_by_angle(0-$offset,$r2); # Get coor's for sequenced 0
                $x1 = $x; $y1 = $y;
                $image -> line($h,$k,$x,$y,$line_color);
                &pos_by_angle(0-$offset-0.07,$r2+50);
                $x2 = $x; $y2 = $y;
                $image -> dashedLine($x1,$y1,$x2,$y2,$black);
                $image -> string(gdGiantFont,$x2-5,$y2-20,"Sequenced Origin",$black);

                &pos_by_angle(3/2*$pi,$r2); $x1 = $x; $y1 = $y; # For replication 0
                &pos_by_angle(3/2*$pi-0.13,$r2+50); $x2 = $x; $y2 = $y;
                $image -> dashedLine($x1,$y1,$x2,$y2,$black);
                $image -> string(gdGiantFont,$x2-173,$y2-10,"Replication Origin",$black);

                &pos_by_angle(1/2*$pi,$r2); $x1 = $x; $y1 = $y; # For replication end
                &pos_by_angle(1/2*$pi+0.06,$r2+60); $x2 = $x; $y2 = $y;
                $image -> dashedLine($x1,$y1,$x2,$y2,$black);
                $image -> string(gdGiantFont,$x2-145,$y2-10,"Replication End",$black);

# Draw concentric circles as log scale, and scale bar (TEXT)
$log_lines = 0;
foreach (0..5) {
        $log_lines += exp(1);
        $concentric[$_] = log($log_lines)/log(6*exp(1)) * $rmax;
}
        $x = $h+$r; $y = $k + $r3 + 30;
        $length = $concentric[$#concentric];                    # Scale bar
        $image -> line($x-$length,$y,$x+$length,$y,$black);
        $image -> line($x,$y-3,$x,$y+3,$black);
        $image -> stringUp(gdLargeFont,$x-8,$y+65,"  50 \%",$black);
        $image -> dashedLine($x,$y-8,$x,$k,$line_color);        # log lines for scale
        $image -> string(gdGiantFont,$x-95,$y+75,"Probability Scale in \%",$black);

foreach (@concentric) {
        $image -> arc($h,$k,($r + $_) * 2,($r + $_) * 2,0,360,$gray20);
        $image -> arc($h,$k,($r - $_) * 2,($r - $_) * 2,0,360,$gray20);
        $image -> line($x+$_,$y-3,$x+$_,$y+3,$black);                   # log lines for scale
        $image -> line($x-$_,$y-3,$x-$_,$y+3,$black);                   # log lines for scale
        $image -> dashedLine($x+$_,$y-8,$x+$_,$k,$line_color);          # log lines for scale
```

```
        $image -> dashedLine($x-$_,$y-8,$x-$_,$k,$line_color);        # log lines for scale
        &p_value($_,$value_times,"1.5f");                             # again, calculates original in %
        $image -> stringUp(gdLargeFont,$x+$_-8,$y+65,$p_value,$black);
        $image -> stringUp(gdLargeFont,$x-$_-8,$y+65,$p_value,$black);
}

# Draw lines for extras, and original values, and send out to file
        open (EXTRAS,">Results/$size/extras_$key\-afinados") || die "Coudn't create file!\n";
        $image -> setBrush($brush3);

        foreach (sort {$a <=> $b} keys %extras) {

          &pos_by_angle($_,$r); $x1 = $x; $y1 = $y;
          &pos_by_angle($_,$r + $extras{$_}); $x2 = $x; $y2 = $y;
          $image -> dashedLine($x1,$y1,$x2,$y2,gdBrushed);
          &p_value($original_extras{$_},$value_times,"1.2e");    #calculate original value in %
          $x3 = ($x1+$x2)/2; $y3 = ($y1+$y2)/2 - 8;

          if ($last ne "" && abs($_ - $last) < 0.05) {       # move overlapped labels
                if ($_ < 1/2*$pi || $_ > 3/2*$pi) {
                        $y3 += 35;
                } else {
                        $y3 -= 35;
                }
          }
          $last = $_;

          $image -> filledRectangle($x3-32,$y3,$x3+35,$y3+15,$white);
          $image -> string(gdLargeFont,$x3-30,$y3,$p_value,$black);

          printf EXTRAS "%5d\t\t$p_value\n",((($_ + $offset) / $factor)/$size +1.5) *
          $extras{$_}/abs($extras{$_}));
        }

# Send to output
if ($key eq "both") {
        print OUT2 $image->png;
} else {print OUT $image->png}

}}

# Subroutines

sub pos_by_coor {
        $x = $_[0];
        $y1 = sqrt($r ** 2 - ($x - $h) ** 2) + $k;
        $y2 = -sqrt($r ** 2 - ($x - $h) ** 2) + $k;
}

sub pos_by_angle {      # Careful, needs radians
        $angle = $_[0];
        $radius = $_[1];
        $y = $radius * sin($angle) + $k;
        $x = $radius * cos($angle) + $h;
}

sub make_poly {
```

```
        local $poly = new GD::Polygon;
        foreach (@_) {
                &pos_by_angle($_,$r);
                $poly -> addPt($x,$y);
        }
        foreach (reverse @_) {
                &pos_by_angle($_,$r + $value);
                $poly -> addPt($x,$y);
        }
        return $poly;
}

sub define_colors {
$b[0] = $white = $image -> colorAllocate(255,255,255);
$b[10] = $black = $image -> colorAllocate(0,0,0);

$b[1] = $gray10 = $image -> colorAllocate(230,230,230);
$b[2] = $gray20 = $image -> colorAllocate(205,205,205);
$b[3] = $gray30 = $image -> colorAllocate(180,180,180);
$b[4] = $gray40 = $image -> colorAllocate(155,155,155);
$b[5] = $gray50 = $image -> colorAllocate(130,130,130);
$b[6] = $gray60 = $image -> colorAllocate(105,105,105);
$b[7] = $gray70 = $image -> colorAllocate(80,80,80);
$b[8] = $gray80 = $image -> colorAllocate(55,55,55);
$b[9] = $gray90 = $image -> colorAllocate(30,30,30);

$c[0] = $red = $image -> colorAllocate(255,0,0);
$c[1] = $orange = $image -> colorAllocate(255,170,0);
$c[2] = $yellow = $image -> colorAllocate(255,255,20);
$c[3] = $green = $image -> colorAllocate(0,255,0);
$c[4] = $darkgreen = $image -> colorAllocate(34,139,34);
$c[5] = $cyan = $image -> colorAllocate(0,255,255);
$c[6] = $blue = $image -> colorAllocate(0,0,255);
$c[7] = $magenta = $image -> colorAllocate(255,0,255);
$c[8] = $purple = $image -> colorAllocate(160,32,240);
$c[9] = $brown = $image -> colorAllocate(130,85,25);

$line_color = $gray30;
$line_color = $gray50;
}

sub draw_all_mets {
# Draws all hits of methylation with a line from center.
        open(MET,"<DB/positions_c") || die "No methylation DB!\n";
        $value = 100;
        while (<MET>) {
                $pos = $_ * $factor - $offset;
                &pos_by_angle($pos,$value);
                $image -> line($x,$y,$h,$k,$yellow);
        }
}

sub p_value {
        # Calculates back original value, NOTE: in % !!
        $p_value = 50 / (exp(1)**($_[0] /$_[1]));
        $p_value = sprintf("%$_[2]",$p_value);
}
```

Programa 10 (lab/get_positions)

```
#!/usr/bin/perl

## Gets positions of a given string from db. 0 exists as 1st position

@strings = ("gatc","cctgg","ccagg");                    # Can't use regexes

@ARGV[0] = "DB/ecoli_line";
die "usage: xtract FILE\n" unless -e ($file = @ARGV[0]);
$db = <>;

open(OUTa,">DB/positions_A") || die "coudn't create log file!\n";
open(OUTc,">DB/positions_C") || die "coudn't create log file!\n";

foreach $string (@strings) {
        while ($where = (index($db,$string,$where))+1)  {
                if ($string =~ /^cc/) {
                        push(@c,($where -1));
                } else {
                        push(@a,($where -1));
                }
        }
        @c = sort {$a <=> $b} @c;
}

foreach (@a) {print OUTa "$_\n"}
foreach (@c) {print OUTc "$_\n"}
```

Programa 11 (lab/voids)

```
#!/usr/bin/perl

## Get largest voids from positions files...


# Open up those files!
open (A, "<DB/positions_A") || die "Can't open the A!\n";
open (C, "<DB/positions_C") || die "Can't open the C!\n";
open (A_OUT, ">Results/voids_A") || die "Can't create voids_A!\n";
open (C_OUT, ">Results/voids_C") || die "Can't create voids_C!\n";
print A_OUT "____\t_____\t___\nSize\tPosition\tEnd\n____\t_____\t___\n";
print C_OUT "____\t_____\t___\nSize\tPosition\tEnd\n____\t_____\t___\n";

# Read files and calculate voids
$last = 0;
while (<A>) {
        chomp;
        $line = $_ - $last -1 . "\t$last\t\t$_\n";
        push(@voids,$line);
        $last = $_;
        }

foreach (sort {$b <=> $a} @voids) {
        print A_OUT;
}

undef @voids;


$last = 0;
while (<C>) {
        chomp;
        $line = $_ - $last -1 . "\t$last\t\t$_\n";
        push(@voids,$line);
        $last = $_;
        }

foreach (sort {$b <=> $a} @voids) {
        print C_OUT;
}

undef @voids;
```

Programa 12 (lab/clusters)

```
#!/usr/bin/perl

## Get largest clusters from positions files... time optimized!
## String instead of vector, and jump by positions files data

# Choose cluster size..
$sizeA = 242;
$sizeC = 385;

# Open up those files!
open (A, "<DB/positions_A") || die "Can't open the A!\n";
open (C, "<DB/positions_C") || die "Can't open the C!\n";
open (A_OUT, ">Results/clusters2_A") || die "Can't create clusters_A!\n";
open (C_OUT, ">Results/clusters2_C") || die "Can't create clusters_C!\n";

print A_OUT "_____\t_____\t_____
Amount\tPosition\tWin size\n_____\t_____\t_____\n";

print C_OUT "_____\t_____\t_____
Amount\tPosition\tWin size\n_____\t_____\t_____\n";

# Should really make the 0&1s now a string!
$blank = "0" x $sizeA;

while (<A>) {
  chomp;
  push(@gatc,$_);
}

$gatc = "0" x 4639221;
substr($gatc,$_,1)="1" foreach (@gatc);

# Now start cycling through only known positions
foreach (1..100) { # Amount of highest returned

$window = $count = $max = $pos_max = 0;

foreach (@gatc) {
  $window = substr($gatc,$_,$sizeA);
  $count = $window =~ s/1/1/g;
  if ($count > $max) {
    $max = $count;
    $pos_max = $_;
  }
}
substr($gatc,$pos_max,$sizeA)=$blank;
print A_OUT "$max\t$pos_max\t\t$sizeA\n";
}
undef @gatc; undef $gatc;

# Should really make the 0&1s now a string!
$blank = "0" x $sizeC;
while (<C>) {
  chomp;
  push(@ccwgg,$_);
```

```
}
$ccwgg = "0" x 4639221;
substr($ccwgg,$_,1)="1" foreach (@ccwgg);

# Now start cycling through only known positions
foreach (1..100) { # Amount of highest returned

$window = $count = $max = $pos_max = 0;

foreach (@ccwgg) {
  $window = substr($ccwgg,$_,$sizeC);
  $count = $window =~ s/1/1/g;
  if ($count > $max) {
    $max = $count;
    $pos_max = $_;
  }
}
substr($ccwgg,$pos_max,$sizeC)=$blank;
print C_OUT "$max\t$pos_max\t\t$sizeC\n";
}
```

Programa 13 (lab/3regions)

```
#!/usr/bin/perl

## Obtains distribution of methylation positions from 3regions db...

$file = "DB/ecoli-3regions";

open (FILE,"<$file") || die "No file!\n";
$db = <FILE>;

open (POSA,"<DB/positions_A") || die "No positions!\n";
open (POSC,"<DB/positions_C") || die "No positions!\n";

foreach $letter ("A","C",) {
        $name = 'POS'.$letter;
        while (<$name>) {
                ${$letter} .= substr($db,$_ +1,"1");
                ${$letter} .= substr($db,$_ +2,"1") if $letter eq "A";
                ${$letter} .= substr($db,$_ +3,"1") if $letter eq "C";
        }
}

foreach $type ("c","t","i","b") {
        ${"countA$type"} = $A =~ s/$type/$type/g;
        ${"countC$type"} = $C =~ s/$type/$type/g;
}

#$countAc += $countAb;
#$countAt += $countAb;
#$countCc += $countCb;
#$countCt += $countCb;

foreach $letter ("A","C",) {
        print "For $letter sites:\n";
        foreach $type ("c","t","i","b") {
                print "$type = ",${"count$letter$type"},"\n";
        }
}
```

Programa 14 (lab/3regions_bias)

```
#!/usr/bin/perl

## Calculate the GC content bias for the 3 regions...

# Define filenames
$regions = "DB/ecoli-3regions";
$line = "DB/ecoli_line";

# Open files and set to variables
open (REG,"<$regions") || die "No file!\n";
$ecoli_sa = <REG>;
open (LINE,"<$line") || die "No file!\n";
$ecoli_line = <LINE>;

# Actually extract the data
foreach (0..length($ecoli_line)-1) {
        $total{substr($ecoli_sa,$_,1)}++;                    # Not needed, just to check
        ${substr($ecoli_sa,$_,1)}{substr($ecoli_line,$_,1)}++;
}

# Add up the b's (to c and t)
foreach (keys %b) {
        $c{$_} += $b{$_};
        $t{$_} += $b{$_};
}
undef %b; undef $total{b}; # Not needed, especially if check removed

# Now the A/T and C/G
foreach ("i","c","t") {
        ${$_}{"A/T"} = ${$_}{a} + ${$_}{t};
        ${$_}{"C/G"} = ${$_}{c} + ${$_}{g};
        $AT = sprintf("%2.2f",${$_}{"A/T"} * 100 / (${$_}{"C/G"} + ${$_}{"A/T"}));
        print "In category $_ there is $AT % A/T\n";
}
```

Programa 15 (lab/total_methylations)

```
#!/usr/bin/perl

## Get type of methylated bases in genome

# Retrieve all info from DB's

        # Get coded info from ecoli.sa
        open (CODED,"<DB/ecoli.s2") || die "No encoded file!\n";
        $coded = <CODED>;
        open (DB, "<DB/ecoli_line") || die "No line!\n";
        $db = <DB>;

foreach $letter ("A","C") {

        # Get 0s&Symbol vector for methylation positions
        undef @{posi.$letter}; ${posi.$letter}[4639220] = 0;
        open(POS,"<DB/positions_$letter") || die "No positions file!\n";

        while (<POS>) {
                if ($letter eq "A") {                   # Getting positions from GATC

                        ${posi.$letter}[$_+1] = ord(substr($coded,$_+1,1)); # Get the A pos
                        ${posi.$letter}[$_+2] = ord(substr($coded,$_+2,1)); # Get the T pos

                } elsif ($letter eq "C") {              # Getting positions from CCWGG

                        ${posi.$letter}[$_+1] = ord(substr($coded,$_+1,1)); # Get the C pos
                        ${posi.$letter}[$_+3] = ord(substr($coded,$_+3,1)); # Get the G pos
                }
        }
}

open(OUT_A, ">Results/gatc_methylations") || die "Can't create file!\n";
open(OUT_C, ">Results/ccwgg_methylations") || die "Can't create file!\n";

undef %counterA;
foreach (@posiA) {
        $counterA{$_}++ unless $_ == 0;
}
undef @posiA; # Free some memory

undef %counterC;
foreach (@posiC) {
        $counterC{$_}++ unless $_ == 0;
}
undef @posiC; # Free some memory

# Now, turn Sensa numbers into something usable...
open (SENSA, "<DB/sensa2/sensa-ttable") || die "No sensa table!\n";

while (<SENSA>) {
        next if ($_ =~ /^#/ || $_ eq "\n");
        @data = split;
        foreach (4..7) {$data[$_] =~ s/^0//}
        $sensa{$data[4]} = "$data[0]\t$data[2]\t$data[1]\t$data[3]";
```

```
        $sensa{$data[5]} = "$data[0]\t$data[2]\t$data[1]\t$data[3]";
        $sensa{$data[6]} = "$data[0]\t$data[2]\t$data[1]\t$data[3]";
        $sensa{$data[7]} = "$data[0]\t$data[2]\t$data[1]\t$data[3]";
}

print OUT_A "SENSA\tHITS\tGEN1\tCOMP\tGEN2\tCOMP\n_____\t____\t___\t____\t____\t____\n";..
$one = $two = $inters = $firsts = $seconds = $thirds = $others = 0;
foreach (sort {$a <=> $b} keys %counterA) {
        ($one,$two,$on,$tw) = split(/\t/,$sensa{$_});
        printf OUT_A "%4d\t%4d\t  %1s\t  %1s\t  %1s\t            %1s\n",$_,$counterA{$_},$one,$on,$two,$tw;
        if ($one eq "-") {
                # do absolutely nothing!
        } elsif ($one == 0) {
                $inters += $counterA{$_}; next; # only works, cause I never have (0 #)
        } elsif ($one == 1) {
                $firsts += $counterA{$_};
        } elsif ($one == 2) {
                $seconds += $counterA{$_};
        } elsif ($one == 3) {
                $thirds += $counterA{$_};
        } else {
                $others += $counterA{$_};
        }
        if ($two eq "-") {
                # do absolutely nothing!
        } elsif ($two == 1) {
                $firsts += $counterA{$_};
        } elsif ($two == 2) {
                $seconds += $counterA{$_};
        } elsif ($two == 3) {
                $thirds += $counterA{$_};
        } else {
                $others += $counterA{$_};
        }
}
# Convert to frequency, per total ccwgg
$int = $inters / 19123;
$fir = $firsts / 19123;
$sec = $seconds / 19123;
$thi = $thirds / 19123;
$oth = $others  / 19123;

printf OUT_A "
Intergenics  = %5d\tor\t%1.6f
First bases  = %5d\tor\t%1.6f
Second bases = %5d\tor\t%1.6f
Third bases  = %5d\tor\t%1.6f
rRNA / tRNA  = %5d\tor\t%1.6f
",$inters,$int,$firsts,$fir,$seconds,$sec,$thirds,$thi,$others,$oth;


print OUT_C "SENSA\tHITS\tGEN1\tCOMP\tGEN2\tCOMP\n_____\t____\t___\t____\t____\t____\n";
$one = $two = $inters = $firsts = $seconds = $thirds = $others = 0;
foreach (sort {$a <=> $b} keys %counterC) {
        ($one,$two,$on,$tw) = split(/\t/,$sensa{$_});
        printf OUT_C "%4d\t%4d\t  %1s\t  %1s\t  %1s\t            %1s\n",$_,$counterC{$_},$one,$on,$two,$tw;
        if ($one eq "-") {
```

```
                # do absolutely nothing!
        } elsif ($one == 0) {
                $inters += $counterC{$_}; next; # only works, cause I never have (0 #)
        } elsif ($one == 1) {
                $firsts += $counterC{$_};
        } elsif ($one == 2) {
                $seconds += $counterC{$_};
        } elsif ($one == 3) {
                $thirds += $counterC{$_};
        } else {
                $others += $counterC{$_};
        }
        if ($two eq "-") {
                # do absolutely nothing!
        } elsif ($two == 1) {
                $firsts += $counterC{$_};
        } elsif ($two == 2) {
                $seconds += $counterC{$_};
        } elsif ($two == 3) {
                $thirds += $counterC{$_};
        } else {
                $others += $counterC{$_};
        }
}

# Convert to frequency, per total ccwgg
$int = $inters / 12042;
$fir = $firsts / 12042;
$sec = $seconds / 12042;
$thi = $thirds / 12042;
$oth = $others  / 12042;

printf OUT_C "
Intergenics  = %5d\tor\t%1.6f
First bases  = %5d\tor\t%1.6f
Second bases = %5d\tor\t%1.6f
Third bases  = %5d\tor\t%1.6f
rRNA / tRNA  = %5d\tor\t%1.6f
",$inters,$int,$firsts,$fir,$seconds,$sec,$thirds,$thi,$others,$oth;
```

Programa 16 (lab/new_methylations)

```
#!/usr/bin/perl

## Calculates all methylations per gen and orders them by gatcs per base

# Open up those files!
open (A, "<DB/positions_A") || die "Can't open the A!\n";
open (C, "<DB/positions_C") || die "Can't open the C!\n";
open (GENES, "<DB/all-genes-ecoli") || die "Can't open genes!\n";
open (OUT, ">Results/methylations_table") || die "Can't create!\n";
print OUT "Mean/None\tCCWGGs\tSize\tStart\tB-number\tName
_________\t_____\t____\t_____\t_______\t____\n";

# Should really make the 0&1s now a string!
while (<A>) {
  chomp;
  push(@gatc,$_);
}
$gatc = "0" x 4639221;
substr($gatc,$_+1,1)="1" foreach (@gatc);

while (<C>) {
  chomp;
  push(@ccwgg,$_);
}
$ccwgg = "0" x 4639221;
substr($ccwgg,$_+1,1)="1" foreach (@ccwgg);

# Read up the GENES file!

while (<GENES>) {
        chomp;
        ($start,$end,$bnumber,$name) = (split(/[\t+\.+]/,$_))[4,6,2,8];
        $start--; $end--; $c_amount = 0;
        $size = $end - $start + 1;
        $c_amount = substr($ccwgg,$start,$size) =~ s/1/1/g;
        $c_amount = "0" unless $c_amount;
        $without = $a_sum = 0;
        foreach ($start..$end) {
                $without++ unless ($a_sum += substr($gatc,$_-1000,2001) =~ s/1/1/g);
        }
        $without *= -1;
        $a_mean = sprintf "%2.4f",$a_sum / $size;
        if ($without) {
                push(@data,"$without\t$c_amount\t$size\t$start\t$bnumber\t$name");
        } else {
                push(@data,"$a_mean\t$c_amount\t$size\t$start\t$bnumber\t$name");
        }
}
foreach (sort {$a <=> $b} @data) {
        print OUT "$_\n";
}
```

Programa 17 (lab/get_bnumbers)

```
#!/usr/bin/perl

## Need b numbers to correlate essential genes in methylation file!

# open up those files!
open (B, "<DB/essential_ecoli") || die "Not essentially so!\n";
open (M, "<Results/methylations_table") || die "No methylations!\n";
open (OUT,">Results/essentials_table") || die "can't create!\n";

# Put bnumbers into array
while (<B>) {
        /(b\d{4})/;
        push(@b_numbers,$1);
}

while (<M>) {
        $b_hash{$1} = $_ if /\t(b\d{4})\t/;
}

foreach (@b_numbers) {
        unless ($b_hash{$_} eq "") {
                push(@essential,$b_hash{$_});
        }
}

foreach (sort {$a <=> $b} @essential) {
        print OUT;
}
```

Programa 18 (lab/genes_in_cluster-voids)

```
#!/usr/bin/perl

## Need to get genes that are included in range specified by files

# Filed needed
$lista = "/home/cei/DB/gabriel/lista-E_coli_K12";
$voidsA = "/home/cei/lab/Results/clusters_voids/voidsA_selected";
$voidsC = "/home/cei/lab/Results/clusters_voids/voidsC_selected";
$clustersA = "/home/cei/lab/Results/clusters_voids/clustersA_selected";
$clustersC = "/home/cei/lab/Results/clusters_voids/clustersC_selected";

# Some kind of double loop?
open (GENES,"<$lista") || die "No genes!\n";
while (<GENES>) {
        push (@genes,[(split /\t|\.{2}|\n/)[4,5,1,7]]); # Start, end, name, description
}
close GENES;

foreach $void ($voidsA, $voidsC) {
        open (IN, "<$void") || die "No $void!\n";
        open (OUT, ">$void-genes") || die "Can't make genes!\n";
        while (<IN>) {
                $flag = "";
                next unless /^\d+/;
                ($size,$vstart,$vend) = (split)[0,1,2];
                print OUT "\n$size\t(VOID)\t$vstart..$vend\n";
                foreach (0..$#genes) {
                        next if ($vstart > $genes[$_][1]);
                        if ($vend < $genes[$_][0]) {
                                $flag++;
                                next;
                        }
                        last if $flag > 10;
                        print OUT "$genes[$_][2]\t$genes[$_][3]\t$genes[$_][0]..$genes[$_][1]\n";
}}}

foreach $cluster ($clustersA, $clustersC) {
        open (IN, "<$cluster") || die "No $cluster!\n";
        open (OUT, ">$cluster-genes") || die "Can't make genes!\n";
        while (<IN>) {
                $flag = "";
                next unless /^\d+/;
                ($size,$vstart,$vend) = (split)[0,1,2]; $vend += $vstart +1;
                print OUT "\n$size\t(CLUSTER)\t$vstart..$vend\n";
                foreach (0..$#genes) {
                        next if ($vstart > $genes[$_][1]);
                        if ($vend < $genes[$_][0]) {
                                $flag++;
                                next;
                        }
                        last if $flag > 10;
                        print OUT "$genes[$_][2]\t$genes[$_][3]\t$genes[$_][0]..$genes[$_][1]\n";
}}}
```