Universidad de Concepción

Facultad de Ingeniería

Departamento de Ingeniería Informática y Ciencias de la Computación

Filtrado de lecturas por abundancia relativa (cuantiles)

Autor:
Ricardo Andrés Charris Jiménez
(2022452901)

Profesora:
Cecilia Hernández Rivas

20 de diciembre de 2025

Índice

1. Introducción

El objetivo de este proyecto es implementar y comparar distintas soluciones para el filtrado de k-mers cuya abundancia relativa sea distinta de los patrones esperados, esto con el fin de estudiar cual es el método más eficiente para realizar la limpieza de los datos.

Para ello, se implementara la estructura de tipo sketch de cuantiles Coooled-KLL propuesta en el paper: **Cooled-KLL: Enhancing Quantile Estimation by Filtering Hot Item**, y se comparara su rendimiento con métodos deterministas como almacenar la distribución de los datos en un vector ordenado, y con el uso de pares {elemento, frecuencia} para disminuir el espacio ocupado por el vector.

Los temas a tratar son los siguientes:

- Descripción de las soluciones existentes, específicamente el uso de un vector ordenado y el uso de un vector ordenado en formato comprimido (pares {elemento, frecuencia}).
- Descripción de la estructura y las operaciones del Cooled-KLL sketch ($quantile(\delta)$ y $rank(x)$).
- Análisis experimental de las estructuras mencionadas en términos de tiempo de construcción, tiempo de consultas, y espacio ocupado.
- Reflexión critica de los resultados obtenidos, y de las ventajas y desventajas de cada estructura.

2. Descripción del problema

2.1. Filtrado de lecturas por abundancia relativa

En los análisis de secuenciación masiva, las lecturas presentan variaciones en la abundancia de sus k-mers o minimizers, este desequilibrio puede sesgar análisis posteriores como la cuantificación o el clustering. El problema consiste en el Filtrado de lecturas por abundancia relativa (Cuantiles), con el objetivo de filtrar lecturas con abundancia anómala de sus k-mers o minimizers, esto se refiere a detectar y descartar aquellas lecturas cuyas abundancias de sus k-mers o minimizers tienen una frecuencia fuera de lo esperado, ya sea demasiado baja o demasiado alta.

2.2. Importancia del problema

El principal área de impacto de este problema es la bioinformática. El filtrado de lecturas de baja calidad es un proceso crucial cuando se trata con datos genómicos, esto debido

a que mejora drásticamente la calidad y precisión de los análisis posteriores al eliminar datos ruidosos o de baja calidad. El filtrado reduce el impacto de errores de secuenciación, duplicaciones y contaminación de los datos, y también, disminuye el tamaño de elementos con los que se trabaja, por lo cual reduce la capacidad de computo y memoria necesaria en etapas posteriores. En resumen, el filtrado es un paso fundamental que transforma datos ruidosos en información interpretable y útil para la ciencia.

3. Soluciones existentes

Actualmente, algunas de las soluciones utilizadas para el filtro de lecturas son las basadas en abundancia absoluta y las basadas en abundancia relativa.

3.1. Umbrales fijos

Uno de los métodos para el filtrado de lecturas consiste en establecer umbrales de abundancia fijos, de esta manera se eliminan aquellas lecturas cuya frecuencia sea menor o mayor a unos limites preestablecidos. El problema de este método, es que no considera la variabilidad en la distribución de los datos ni su tamaño, distintos datasets requerirán distintos umbrales debido a la variación en su distribución y en su tamaño. Esto conlleva a que para que el método sea efectivo sea necesario un estudio previo de la repartición de los datos.

3.2. Distribución de frecuencias

Este método consiste en calcular la distribución exacta de la abundancia de los k-mers o minimizers en las lecturas, a partir de la distribución se puede aplicar un filtrado a los datos en base a cuantiles. El problema con el uso de métodos que calculen la distribución exacta radica en el uso de memoria que conlleva almacenar la totalidad de los datos en memoria, además, al ingresar un nuevo dato es necesario volver a calcular la distribución o insertar el dato en su posición correspondiente realizando búsqueda.

4. Cooled-KLL sketch

En esta sección se detalla la estructura y funcionamiento del Cooled-KLL sketch. Esta estructura esta compuesta de dos partes, el Hot filter y el KLL sketch tradicional.

4.1. KLL sketch

El KLL sketch esta compuesto por una serie de H compactadores, el compactador más grande corresponde al del nivel más alto y tiene un tamaño k, el tamaño de los compac-

tadores inferiores esta determinado por la siguiente ecuación: $k * c^{H-h}$ (pero mínimo 2) ; $c \in (0.5, 1)$, con H el nivel mayor del sketch y h el nivel del compactador en cuestion. Cada nivel tiene asignado un peso dado por 2^h, el cual determina a cuantos elementos representa la presencia de un ítem en dado nivel.

A continuación se describe el procedimiento de inserción de datos en el sketch, el calculo de cuantiles y el calculo del rank.

- **Insert(x):** Al insertar un elemento en el sketch, este se inserta en el compactador de nivel 0, si el compactador alcanza el máximo de su capacidad, se inician un conjunto de compactaciones en los niveles que sea necesario. Las compactaciones consisten en elegir aleatoriamente y con igual probabilidad los elementos pares o impares en el nivel, los elementos elegidos se transfieren al nivel superior y el resto son eliminados. Este proceso se realiza iterativamente hasta que no haya que transferir elementos o hasta que se cree un nuevo compactador si se llega al nivel superior.

- **Quantile(δ):** Para realizar esta consulta se obtienen las frecuencias de todos los elementos en cada uno de los compactadores y con esos datos se arman dos arreglos, uno con los elementos ordenados y otro con sus frecuencias. Luego, se recorre el arreglo de frecuencias hasta encontrar el ítem que equivale al indice $\delta * N$, N el total de elementos, finalmente, se retorna el ítem del arreglo de los elementos en la posición encontrada en el arreglo de frecuencias.

- **rank(x):** Para realizar esta consulta, se recorren todos los compactadores y se cuentan (considerando los pesos) cuantos ítems son menores o iguales a x. Finalmente, se retorna el total de ítems contabilizados.

4.2. Hot filter

El Hot filter consta de un arreglo de W bloques, los cuales a su vez contienen un conjunto de E entradas tal que cada entrada almacena un par {clave, valor}. Cada uno de los bloques contiene un campo *voto*, el cual es utilizado para determinar que elementos permanecen en el Hot filter y que elementos son relegados al KLL. Para determinar a que bloque pertenece un elemento se utiliza una función hash para indexar los bloques, de esta manera se determina rápidamente la posición de cada elemento.

4.3. KLL + Hot filter

A continuación se explica como se realizan las operaciones insert(x), quantile(δ) y rank(x) en el Cooled-kll sketch.

- **Insert(x):** La inserción de elementos se divide en 4 casos:

- **Caso 1:** Si el elemento x entrante ya se encuentra en alguna entrada del bloque al que pertenece, incrementamos la frecuencia del elemento en 1.

- **Caso 2:** Si el elemento x no se encuentra en ninguna entrada del bloque al que pertenece, pero, hay alguna entrada vacía. Insertamos el elemento con frecuencia 1 en esa entrada.

- **Caso 3:** Si el elemento x no se encuentra en ninguna entrada del bloque y no hay ninguna entrada disponible. Se aumenta el valor del *voto* en 1, luego, se verifica la siguiente inecuación $\frac{voto}{fmin} > \lambda$, donde $fmin$ es la frecuencia mínima en el bloque y λ es el eviction threshold, parámetro usado para decidir cuando enviar un elemento al KLL. Si la inecuación se cumple, el elemento x se inserta en el KLL-sketch.

- **Caso 4:** Si el elemento x no se encuentra en ninguna entrada del bloque y no hay ninguna entrada disponible. Se aumenta el valor del *voto* en 1, luego, se verifica la siguiente inecuación $\frac{voto}{fmin} <= \lambda$. Si la inecuación se cumple, el elemento con la frecuencia mínima es desplazado al KLL-sketch, para esto se inserta una cantidad de veces igual a su frecuencia asignada. Finalmente, el elemento entrante se inserta con frecuencia 1 en la entrada que quedo vacía y el *voto* se resetea a 0.

  La inserción del elemento relegado al sketch se puede optimizar para no tener que insertarlo una cantidad de veces igual a su frecuencia y evitar provocar tantas compactaciones. Para esto se calcula la factorización en potencias de dos de la frecuencia del ítem, y se inserta el ítem en cada nivel que corresponda.

- **Quantile(δ):** Se calcula idénticamente que en el KLL-sketch, solo que adicionalmente considerando los elementos del Hot filter.

- **rank(x):** Se calcula el rank(x) del KLL-sketch y se le suma la cantidad de elementos menores o iguales a x en todas las entradas del Hot filter (considerando sus frecuencias).

5. Resultados experimentales

En esta sección se explica el procedimiento realizado para evaluar y comparar el rendimiento del Cooled-KLL sketch con las estructuras antes mencionadas, el vector ordenado, y el vector de pares {elemento, frecuencia}.

5.1. Dataset

Para la realización de los experimentos se utilizó el dataset 50 genomas de Escherichia Coli, el cual contiene 50 secuencias genomicas de esta bacteria. El dataset está dividido en 50 archivos y pesa un total de 176 MB.

5.2. Entorno de ejecución

Los experimentos se realizaron en una máquina con un procesador intel core i7 11800-H con frecuencia base de 2.3 GHz y una frecuencia de turbo máxima de 4.6 GHz, el equipo tiene 16 GB de memoria RAM DDR4 a 3200 MHz.

5.3. Experimentos

- **Estimación de distribución** A partir del dataset especificado se obtuvieron k-mers de largo = $\{5, 6, 7, 8, 9, 10, 11, 12, 15, 21, 26, 31\}$ y se contabilizaron sus frecuencias. Luego, se insertaron las frecuencias de los k-mers en sketches construidos con distintas configuraciones y se obtuvo la distribución de frecuencias estimada almacenando el resultado de la consulta quantile(δ), con $\delta \in [0, 0.001, 0.002, ..., 0.998, 0.999, 1]$, adicionalmente, se almaceno la distribución real de los datos, la estimación del rank para cada uno de los resultados de la operación quantile, el rank real y el espacio usado por cada una de las soluciones. Las configuraciones usadas para los sketches fueron:

  1. **Configuración 1:** 100 bloques en el Hot filter, 10 entradas por bloque, tamaño del compactador más grande igual a 100.
  2. **Configuración 2:** 10 bloques en el Hot filter, 5 entradas por bloque, tamaño del compactador más grande igual a 20.
  3. **Configuración 3:** 15 bloques en el Hot filter, 5 entradas por bloque, tamaño del compactador más grande igual a 20.'
  4. **Configuración 3:** 20 bloques en el Hot filter, 5 entradas por bloque, tamaño del compactador más grande igual a 20.

  Cabe destacar que todas las configuraciones usaron un eviction threshold de 16 y un factor de compresión igual a 0.7.

  Los datos se almacenaron en dos CSV, uno con datos sobre las operaciones de la estructura y otro con datos sobre la memoria utilizada.

- **Experimento construcción**
  Se midió el tiempo que se tardaba en construir cada una de las estructuras, el Cooled-

KLL sketch, el vector ordenado, y el vector comprimido de pares; esto utilizando k-mers de largo entre 3 y 30, aumentando el k en 3 en cada iteración. Para este experimento se utilizó la configuración 1 del sketch.

- **Experimento consultas**
  Se midió el tiempo utilizado para realizar las consultas quantile(δ) y rank(x), para rank se ocupó cada valor obtenido de la consulta quantile(δ), y para quantile se ocuparon $\delta \in [0, 0.001, 0.002, ..., 0.998, 0.999, 1]$. Para efectos de los gráficos generados, se utilizaron distintas configuraciones de sketch dependiendo de con cual k se iba a realizar el experimento.

- **Filtrado de lecturas**
  Se estimó la distribución de frecuencias usando la configuración 1 del KLL-sketch para k-mers de largo entre 3 y 30, iterando de 3 en 3. Luego de estimada la distribución, se estimaron los cuantiles 0.01 y 0.99, como resultado se obtuvo el límite izquierdo y el límite derecho, todos los k-mers con frecuencia menor al límite izquierdo o mayor al límite derecho son eliminados. Finalmente, se almacenó la cantidad de k-mers distintos y la cantidad total que fueron eliminados.

5.4. Resultados

- **Uso de memoria**
  Para comparar el uso de memoria de cada una de las soluciones para los diferentes k y las diferentes configuraciones del sketch se crearon 4 tablas con la información corespondiente.

Resumen de Memoria por K-mer (KB)
(NB_100_BC_10_CS_100)

K-mer	Elements	Unique Elem.	Sketch Mem (KB)	Vector Mem (KB)	Comp. Vector Mem (KB)
5.0	512.0	512.0	20.61	4.02	16.02
6.0	2080.0	2061.0	29.12	16.27	64.43
7.0	8192.0	7390.0	29.3	64.02	230.96
8.0	32896.0	11915.0	29.95	257.02	372.37
9.0	131072.0	7644.0	29.62	1024.02	238.9
10.0	524800.0	4421.0	30.1	4100.02	138.18
11.0	2091856.0	2500.0	29.39	16342.65	78.15
12.0	7784369.0	1388.0	28.93	60815.41	43.4
15.0	51349006.0	368.0	18.41	401164.13	11.52
21.0	62447043.0	226.0	16.19	487867.55	7.09
26.0	62961837.0	191.0	15.64	491889.38	5.99
31.0	63348389.0	158.0	15.12	494909.31	4.96

Figura 1: Uso de memoria (configuración 1)

La tabla muestra la cantidad de elementos y elementos unicos, la memoria en KB usada por el sketch, el vector ordenado y el vector comprimido. En primer lugar,

es necesario definir que 'elements' hace alusión a cuantos k-mers distintos se encontraron en el dataset, mientras que 'Unique Elem.' se refiere a cuantas frecuencias distintas fueron encontradas en el dataset, por ejemplo:

Suponinedo que tenemos los siguientes k-mers: {ACG,CGT,TAA,ACG,TAA}, la cantidad de elementos sería 3, dado que hay 3 k-mers distintos, sin embargo, la cantidad de elementos únicos sería 2, ya que hay 2 k-mers con frecuencia 2 y 1 k-mer con frecuencia 1, por lo tanto, solo hay 2 frecuencias distintas.

La información de la tabla corresponde al sketch de la configuración 1, se puede notar que esta configuración es viable para k-mers de largo entre 7 y 12, ya que para estos largos la memoria usada por el sketch es menor a las otras dos soluciones. También se puede notar que la ganancia en memoria aumenta a medida que hay una mayor cantidad de elementos únicos en relación a la cantidad total de elementos, en el mejor caso se ve una reducción de aproximadamente 9 veces el espacio ocupado de las otras soluciones.

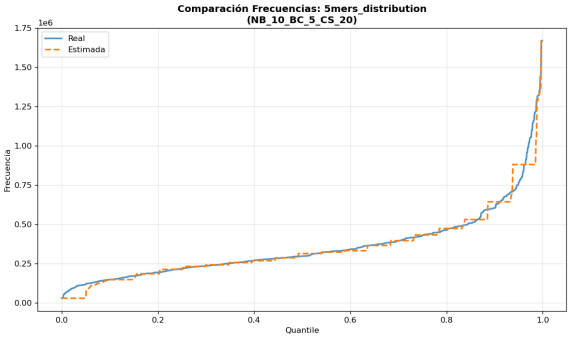
Resumen de Memoria por K-mer (KB)
(NB_10_BC_5_CS_20)

K-mer	Elements	Unique Elem.	Sketch Mem (KB)	Vector Mem (KB)	Comp. Vector Mem (KB)
5.0	512.0	512.0	2.48	4.02	16.02
6.0	2080.0	2061.0	2.59	16.27	64.43
7.0	8192.0	7390.0	2.7	64.02	230.96
8.0	32896.0	11915.0	2.79	257.02	372.37
9.0	131072.0	7644.0	2.72	1024.02	238.9
10.0	524800.0	4421.0	2.82	4100.02	138.18
11.0	2091856.0	2500.0	2.84	16342.65	78.15
12.0	7784369.0	1388.0	2.87	60815.41	43.4
15.0	51349006.0	368.0	2.74	401164.13	11.52
21.0	62447043.0	226.0	2.63	487867.55	7.09
26.0	62961837.0	191.0	2.59	491889.38	5.99
31.0	63348389.0	158.0	2.62	494909.31	4.96

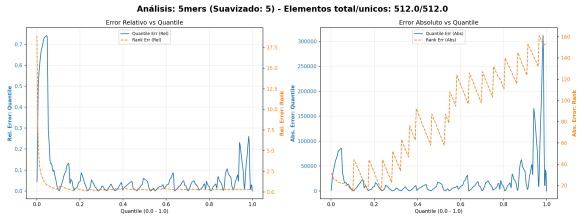
Figura 2: Uso de memoria (configuración 2)

Para los tamaños de k-mer mayores a 12 se utilizo una configuración de sketch más pequeña, de esta manera se logra utilizar una menor memoria que las otras soluciones y se hace viable el estudio del error y las consultas.

- **Error consultas**
  A continuación se muestran gráficos de los errores relativos y absolutos asociados a las consultas quantile(δ) y rank(x) para distintas configuraciones del Cooled-KLL.



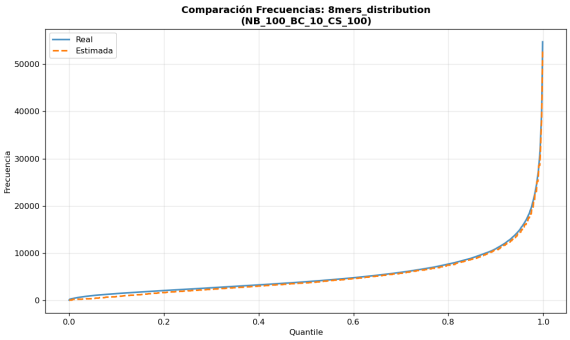
(a) Distribución estimada vs real



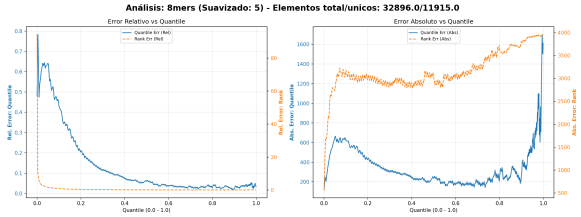
(b) Error relativo y absoluto

Figura 3: Resultados para 5-mers (configuración 2)

Podemos notar visualmente que la distribución estimada por el sketch se adapta a la distribución real de mejor manera en las zonas intermedias, en los extremos hay un mayor alejamiento de la curva. En cuanto al error, podemos notar que el error relativo para la operación quantile() es bastante alto en los extremos, se vuelve aceptable a partir del cuantil 0.1 y hasta el cuantil 0.9 aproximadamente. Para la operación rank() el error no es muy notorio a excepción de en los primeros cuantiles.



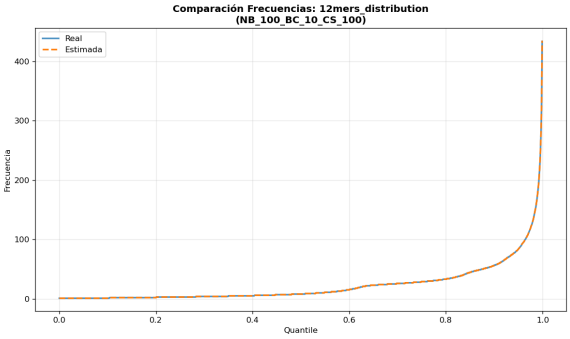
(a) Distribución estimada vs real



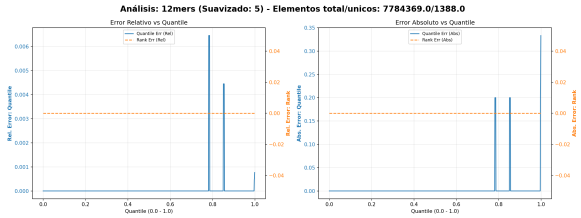
(b) Error relativo y absoluto

Figura 4: Resultados para 8-mers (configuración 1)

Para este sketch es posible notar visualmente que hay una adaptación mucho mejor a la distribución real, sin embago, si nos fijamos en los errores, a partir del cuantil 0.3 notamos que el error relativo baja de 0.1, a pesar de usar el sketch más grande el error es bastante alto en los inicios, esto se debe a la gran cantidad de elementos únicos que hay para k=8, dado que la memoria usada por este sketch es varias veces más baja que los otros métodos, podríamos reducir el error al hacer un sketch más grande.



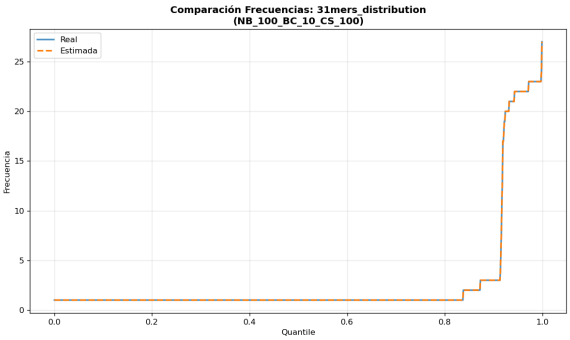
(a) Distribución estimada vs real



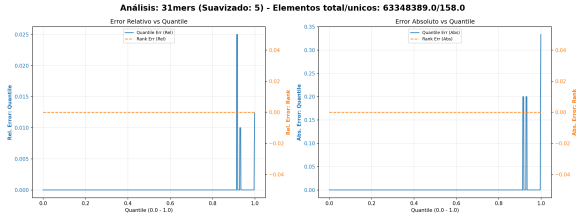
(b) Error relativo y absoluto

Figura 5: Resultados para 12-mers (configuración 1)

Para 12-mers es posible notar que la distribución estimada por el sketch se adapta casi perfectamente a la distribución real. Los errores relativos de las operaciones para casi todos los cuantiles son 0 a excepción de algunos pocos que son despreciables. En general, esta configuración de sketch para este largo de k-mer minimiza el error de las operaciones, dado al bajo error se podría disminuir el tamaño del sketch para ganar espacio a cambio de exactitud.



(a) Distribución estimada vs real



(b) Error relativo y absoluto

Figura 6: Resultados para 31-mers (configuración 1)

Para 31-mers se utilizó el sketch con la configuración 2, debido a que hay pocos elementos unicos por lo tanto es necesario reducir el espacio del sketch para que sea viable. En el gráfico se puede notar que los errores son prácticamente 0 para todos los cuantiles, notandose algunas elevaciones en los cuantiles más grandes.

- **Tiempo de construcción**

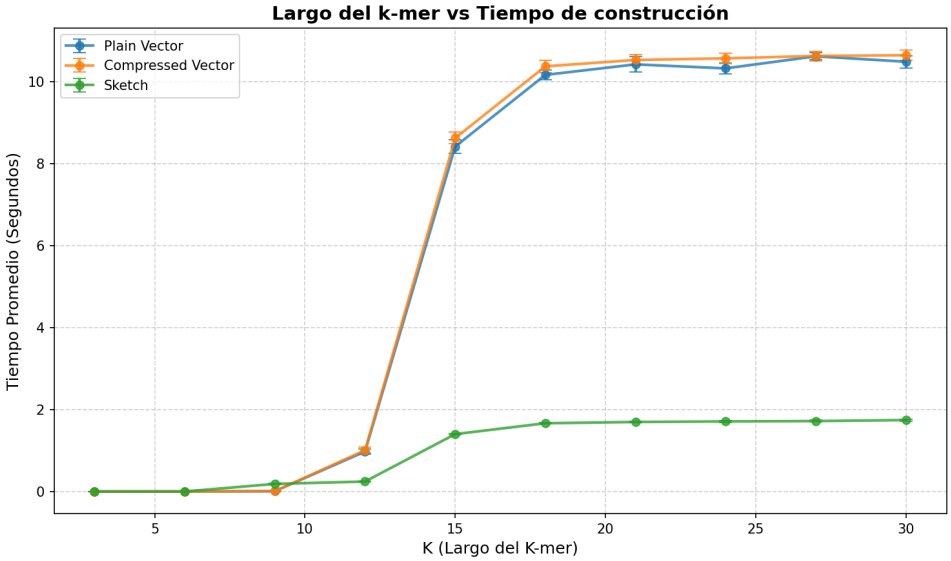


Figura 7: Tiempo de construcción

Como se ve en la figura, el Cooled-KLL sketch tarda menos tiempo en construirse a partir de todos los elementos que las otras dos soluciones, esto se debe a que tanto el vector ordenado como el vector comprimido requieren ordenar los datos para ser construidos, en cambio, el sketch solo tiene que insertarlos y para los casos donde hay pocos elementos únicos esto toma tiempo constante amortizado por cada inserción.

- **Tiempo de consultas**

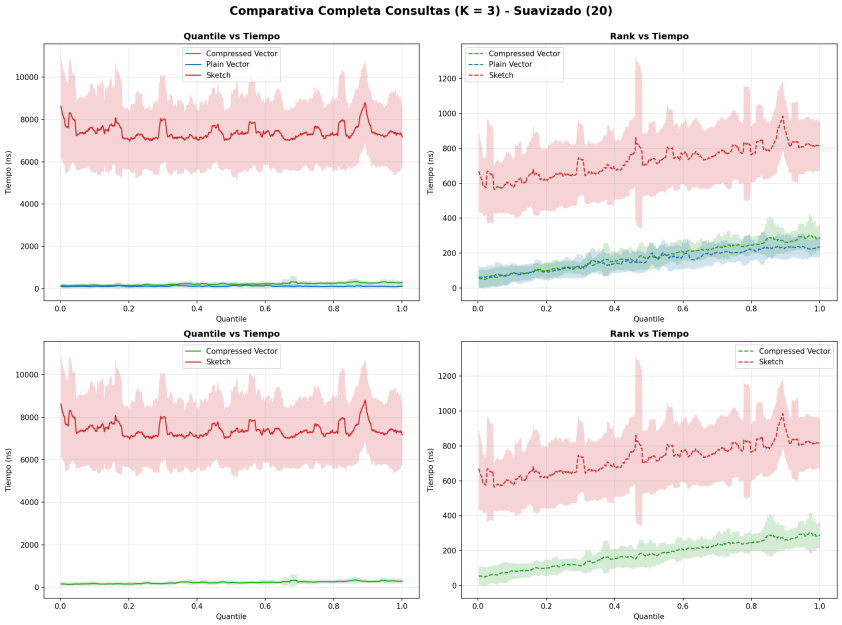


Figura 8: Tiempo de operaciones

Se puede notar que tanto el vector ordenado como el vector comprimido superan ampliamente el tiempo utilizado para realiar tanto consultas quantile() como consultas rank(), por lo tanto, usar un sketch para un tamaño de k-mer tan pequeño no es viable. Además, el espacio utilizado por el vector ordenado es pequeño, por lo cual el sketch no tiene ninguna ventaja en este caso.

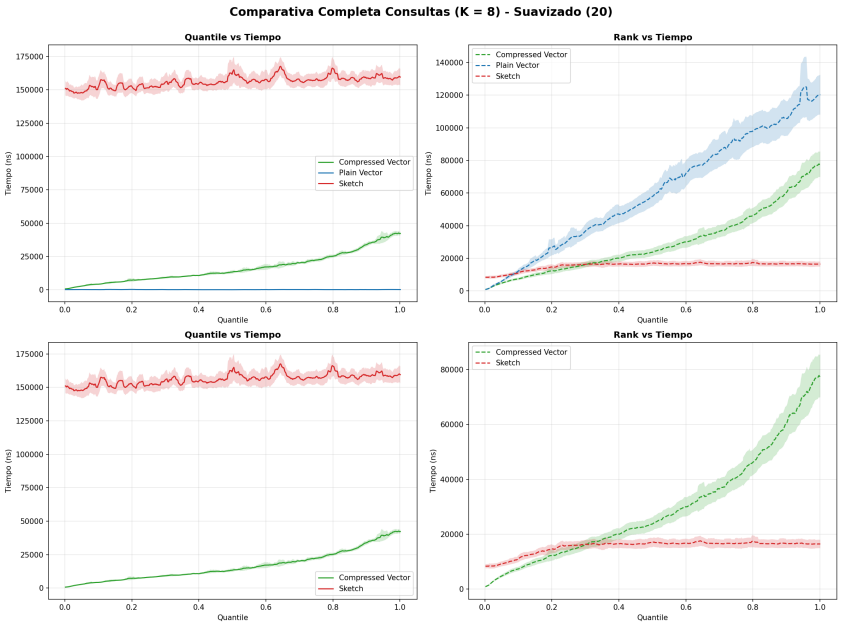


Figura 9: Tiempo de operaciones

Para este caso, 8-mers, se puede notar que el sketch tarda menos tiempo en realizar la operación rank() que los otros métodos, esto es porque para los k-mers de largo 8 se encontraron muchos elementos únicos y los vectores tienen que recorrer linealmente cada uno de estos. En cuanto a la operación quantile(), se puede notar que el vector comprimido tarda entre 3 a miles de veces menos que el sketch, el tiempo se reduce considerablemente para los primeros cuantiles ya que tiene que recorrer menos elementos.

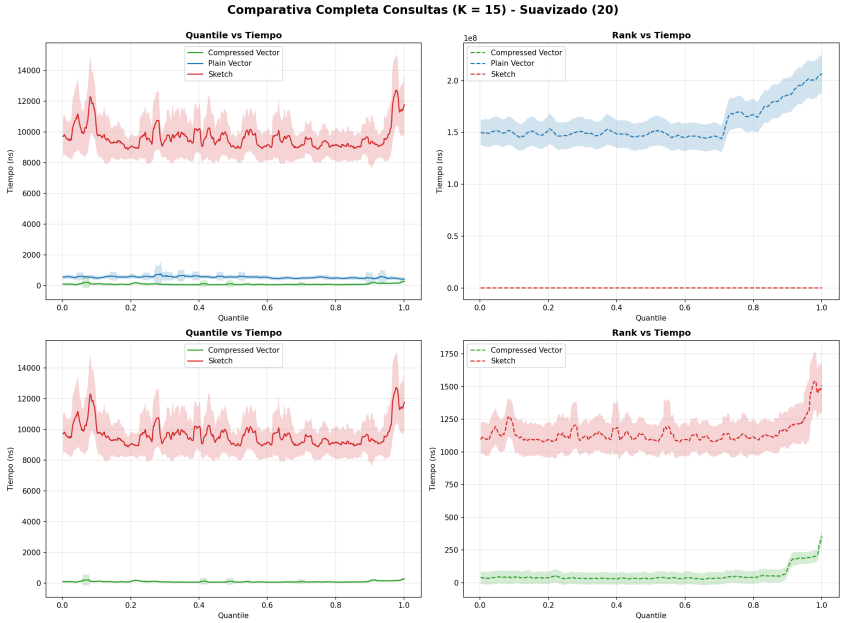


Figura 10: Tiempo de operaciones

Finalmente, para k-mers de largo 15, el uso de un vector ordenado se vuelve inviable, ya que tiene que recorrer muchos elementos para obtener los resultados, además, como se vio previamente ocupa una gran cantidad de espacio. Se puede ver que el sketch sigue siendo la estructura más lenta para responder ante las consultas de tipo quantile().

En resumen, se puede comprobar que el sketch no aporta en terminos de velocidad de consulta, exceptuando en la operación rank para datasets con muchos elementos únicos, aunque la operación utilizada para el filtrado de lecturas es quantile(), por lo tanto no es una ventaja relevante.

- **Filtrado de lecturas**

Filtrado k-mers presentes en 50 genomas de Escherichia Coli

K	Low Q	Up Q	Low Bound	Up Bound	Elements	Unique Elim.	Total Elim.
5	0.01	0.99	29,533	676,732	181,134,171	39	36,667,905
8	0.01	0.99	625	27,005	181,129,479	965	13,374,399
10	0.01	0.99	14	2,539	181,128,230	8,090	12,638,970
12	0.01	0.99	1	180	181,127,297	77,039	22,294,682
15	0.01	0.99	1	25	181,126,179	430,038	16,425,213
18	0.01	0.99	1	23	181,125,253	207,108	6,067,320
21	0.01	0.99	1	23	181,124,393	160,972	4,680,734
26	0.01	0.99	1	23	181,123,040	150,831	4,335,470
31	0.01	0.99	1	23	181,121,741	145,624	4,151,774

Figura 11: Filtro de lecturas

Ya que se filtraron los k-mers cuya abundancia se encuentre inferior al cuantil 0.01 o superior al cuantil 0.99, en total se deberían haber eliminado aproximadamente un 2 % de los datos, aunque esto depende de como sea la distribución de elementos, si hay muchos elementos en los extremos de la distribución con igual abundancia puede pasar que se eliminen una gran cantidad de elementos debido a esto, por ejemplo, el caso de k=5 que es donde se eliminan la mayor cantidad de elementos, un 20 % del total.

Cabe destacar que anteriormente se vio que el sketch tiende a errar más en los cuantiles extremos y como el problema objetivo consiste en filtrar en base a percentiles extremos, el sketch posee un gran problema en este aspecto. Sin embargo, también se concluyo que aumentando el tamaño del sketch se puede reducir el error, entonces hay formar de arreglar este defecto.

6. Conclusiones

En conclusión, a partir de los experimentos realizados podemos decir que utilizando una configuración de parámetros adecuada es posible reducir en gran medida el espacio utilizado en comparación al vector ordenado y al vector comprimido. La estructura es dinamica, ya que sigue permitiendo realizar las consultas a pesar de que se inserten nuevos elementos, lo cual es beneficioso si se trabaja con flujos de datos continuos. Por otro lado, en datasets con muchos elementos únicos, la consulta de rank es muy eficiente y puede igualar e incluso superar la velocidad de los otros dos métodos.

En cuanto a los aspectos negativos, determinar el tamaño optimo del sketch para los datos con los que se va a trabjar requiere un estudio del dataset y/o la realización de pruebas con distintas configuraciones para encontrar la más beneficiosa, es decir, la que ocupe

menos espacio y entregue un error aceptable, si no se ocupa una configuración buena, el espacio ahorrado puede ser poco o nulo, o el error en las consultas puede ser muy alto. En distribuciones con pocos datos únicos no ahorra mucho espacio en comparación al vector comprimido, y considerando que este realiza las operaciones más rápido, en esos casos no sería recomendable usar el sketch. Finalmente, el error tiende a masificarse en los extremos, lo que es un obstaculo grave para el problema planteado, ya que se busca filtrar los datos en base a los cuantiles extremos.

A modo de resumen, la estructura Cooled-KLL puede ser una herramienta muy útil para algunos conjuntos de datos, tiene una gran fortaleza en cuanto a eficiencia espacial, siempre y cuando las condiciones lo permitan. Sin embargo, hay casos en donde utilizar un sketch es inviable o ineficiente, y será mejor recurrir a las estructuras deterministas.

7. Referencias

1. Github proyecto
2. Paper "Cooled-KLL: Enhancing Quantile Estimation by Filtering Hot Item"
3. EFECTO DEL FILTRADO DE SECUENCIAS EN EL ENSAMBLADO DEL GENOMA DE Bacillus altitudinis 19RS3 AISLADO DE Ilex paraguariensis