Centro de Investigación en Matemáticas (CIMAT). Unidad Monterrey

Ciencia de datos (Tarea 2)

José Antonio Garcia Ramirez

1 de abril de 2018

1. Ejericicio 1

Considera los datos del archivo ***gene_expression_2classes.csv****, que contiene la expresíon genética de 1000 genes en 40 muestras de tejido, de las cuales las primeras 20 son de pacientes sanos y las 20 restantes de pacientes enfermos de alguna enfermedad.*

1. *Compara los métodos de clustering jerárquico y los basados en k-means en los datos. ¿Puedes identificar los dos grupos existentes? Prueba usando medidas de disimilaridad euclidena y correlacíon, así como diferentes tipos de enlace. ¿Cómo cambian los resultados realizando PCA previamente a los datos? ¿Cuántos componentes sugieres usar?*
   Realiza un reporte breve de todos estos aspectos, ilustrando de forma adecuada tus hallazgos y conclusiones.

   Se comenzó transponiendo los datos, es decir se partió de un conjunto de datos con 40 observaciones (pacientes) y 1000 variables (respuestas de los genes). Al realizar k-means sobre el conjunto de datos mencionado, conociendo de manera a priori que se dispone de dos grupos no me fue posible identificar los dos grupos, hubo iteraciones en donde esto se acercaba, pero fallaba en uno o más individuos. Se procedió a realizar k-means sobre los datos escalados (a cada columna se resto su media y se dividió por su desviación estándar correspondiente). Con lo cual se obtiene una identificación correcta y 100 % precisa de los dos grupos de pacientes. El resultado

sin embargo de este k-means, es poco estable pues depende de la inicialización inicial de los centroides, pero en promedio la implementación del kernel base de R, logro la separación de los dos grupos. En la figura 1.1 se muestra la clasificación de k-means sobre los datos escalados, además el conjunto de datos se proyecta sobre las dos primeras componentes principales, que pese a solo representar poco más del 11 % de la varianza total, hacen explicito que la primer componente principal basta para separar ambos grupos.

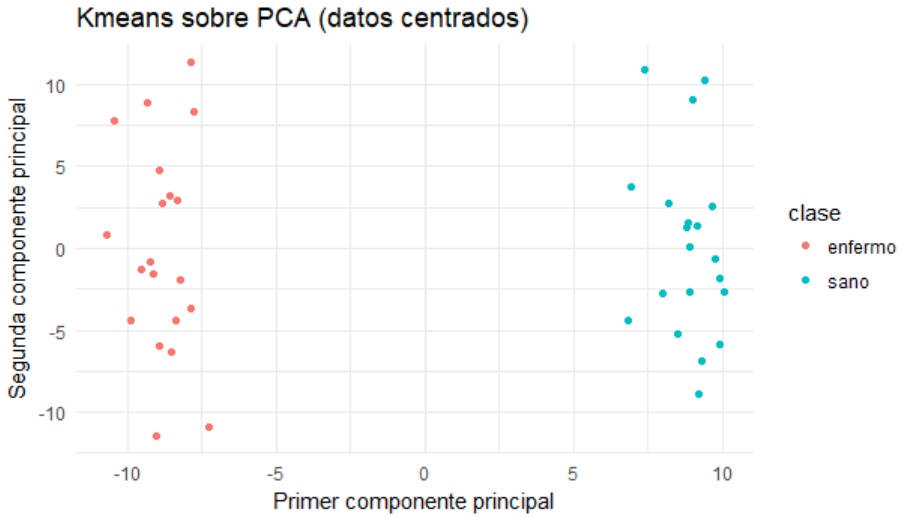

Figura 1.1: Resultado promedio de k-means sobre el conjunto de datos escalado, proyectado en las dos primeras componentes, considerando PCA con una matriz de correlación.

Siguiendo con el ejercicio se encontró que los métodos jerárquicos sobre la matriz de distancias (sobre los datos escalados) considerando la métrica euclidiana y un link de 'war.D' logran la misma separación que se obtuvo en el punto anterior, en la figura 1.2 se representa esta clasificación que también se obtiene considerando la similaridad inducida por la disimilaridad que representa la matriz de correlación de nuestro conjunto de datos y el mismo método de 'war.D'.

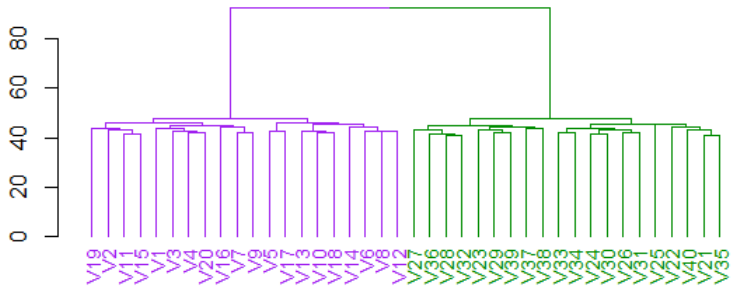

Figura 1.2: Conjuntos de pacientes resultado de usar un algoritmo de clúster jerárquico ('war.D') con una similaridad dada por una distancia euclidiana sobre los datos escalados.

2. *Uno de los médicos quisiera saber qué genes muestran mayores diferencias entre los dos grupos de tejido. ¿Cómo lo verificarias? Implementa tu idea.*

   La primera idea fue utilizar un heatmap sobre el conjunto de datos, definido en el ejercicio anterior, el cual se ilustra en la figura 1.3 pero se reporta sin los dendogramas que efectúa la implementación pues lo importante de esta figura es que exhibe que sí existe un conjunto de genes que se diferencia de los demas.

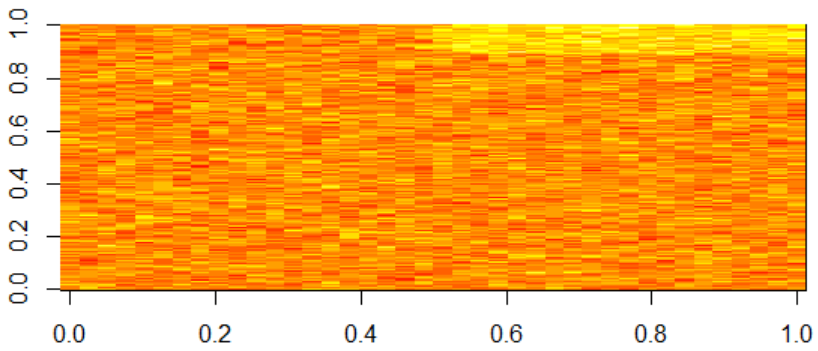

Figura 1.3: Mapa de calor de los datos escalados. Las filas de la matriz corresponden a genes en nuestro conjunto de datos, y las columnas a pacientes. Se nota que existe una agrupación entre los genes (esquina superior derecha).

Entonces se procedió a calcular la matriz de correlaciones de nuestro conjunto de datos señalado en el inciso anterior con lo que se obtiene una matriz de 1000×1000, que consideramos como una matriz de disimilaridad con la que construimos una similaridad (restándola a la matriz unidad de dimensión 1000) y sobre la cual se efectuó un método de clúster jerárquico (también utilizando el método de 'war.D') con lo que identificamos los dos grupos de genes reportados en la figura 1.4, de los cuales el grupo identificado en color morado y con cardinalidad menor enlistado es el siguiente (en correspondencia con el número de registro del archivo .csv original): 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 156, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531,532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550,551, 552, 553, 554, 555, 556, 557, 558, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570, 572, 574, 575, 576, 577, 578, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 900, 920 y 967.

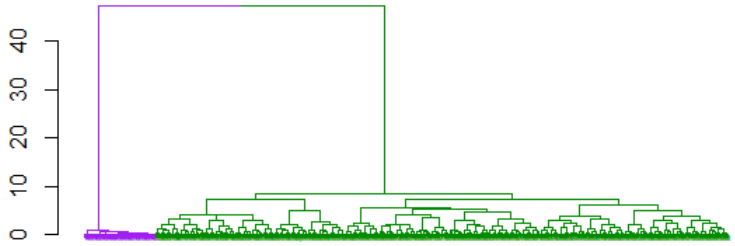

Figura 1.4: Conjuntos de genes resultado de usar un algoritmo de clúster jerárquico (‘war.D’) con una similaridad inducida por la correlación entre genes en la muestra.

La lista anterior constituye nuestro conjunto de genes importantes, pues de hecho al excluirlos del análisis y realizar PCA sobre los datos restantes (y escalados), se obtiene la representación de los datos mostrada en la figura 1.5 en la cual vemos que la distinción entre ambos grupos de pacientes deja de ser sencilla, además ahora las dos primeras componentes principales solo explican poco más del 7 % de la varianza total.

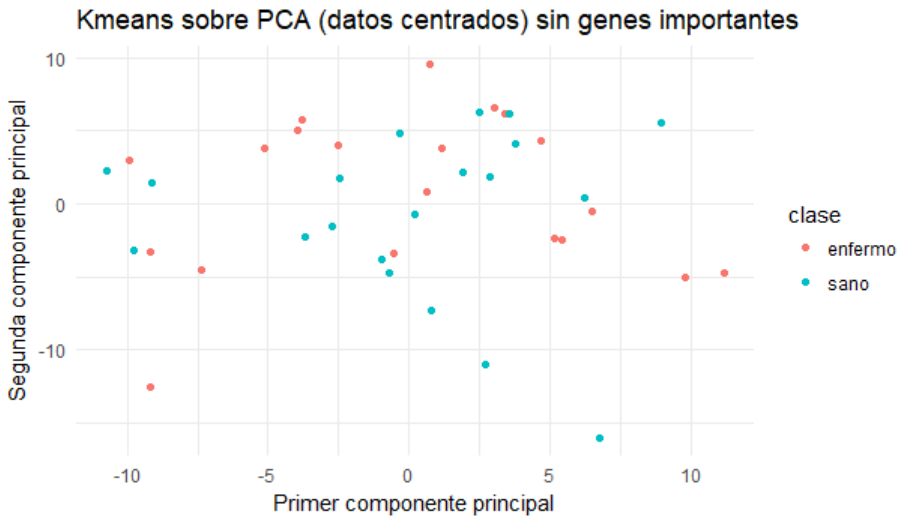

Figura 1.5: Conjunto de datos original (sin los genes que enlistamos) proyectado en las dos primeras componentes, considerando PCA con una matriz de correlación. Nótese que la distinción entre grupos deja de ser sencilla a diferencia de la representación que se obtiene con este conjunto de genes.

Para concluir el código de este análisis se reporta en el anexo A al igual que en el archivo ejercicio1.R

2. Ejercicio 2

Implementa kernel k-means basandote en el artículo de Inderjit Dhillon, Yuqiang Guan and Brian Kulis. A Unified view of Kernel k-means, Spectral Clustering and Graph Cuts. UTCS Technical Report, 2005
Escoge (o genera) algunos conjuntos de datos adecuados para verificar la eficiencia y ventajas del método. Comparalo con otros métodos para mostrar en qué casos es mejor su desempeño.
Basandome en el paper mencionado mi implementación se base en el algoritmo marcado como Algorithm 1, considerando todos los pesos como la unidad, el algoritmo mencionado para Weighted Kernel k-means se reduce al algoritmo Kernel k-means, sin embargo consideré en mi implementación los comentarios señalados en la sección 4.4 (Enforcing Positive Definiteness) e incorpore la idea del shift en mi implementación. Anexo el codigo de mi implementación en el anexo B, así como en un archivo .R llamado 'kernel_kmeans' (que también incluye la comparación con 'kmeans' del kernel base de R y la implementación de 'kkmeans' del package 'Kernlab', que a su vez se basa en el mismo paper, es importante señalar que la implementación 'kkmeans', es superior a la mía en cuanto a accuracy y

tiempos de ejecución como lo reportó en los siguientes párrafos).
Para comenzar genere una muestra parecida a la del paper la cual consiste en 400 observaciones de las cuales las 200 primeras corresponden a una circunferencia con centro en $(0{,}5, 0{,}5)^t$ y de radio 1 a las cuales en cada componente agregue ruido con distribución N(0,1/10), las segundas 200 observaciones corresponden a una circunferencia con el mismo centro pero con radio 4, también en cada componente se agregó ruido con la misma distribución. En la figura 2.11 se muestran esta muestra aleatoria generada.

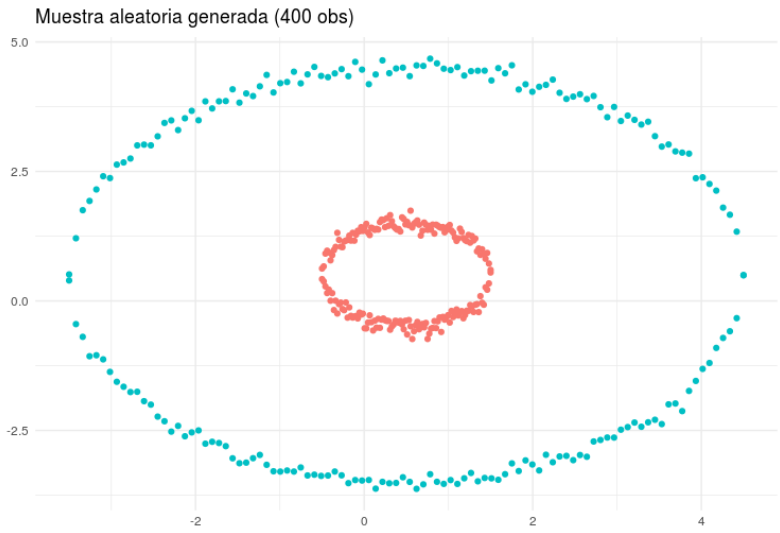

Figura 2.1: Conjunto de datos generado para comparar la implementación de Kernel k-means, nótese la no linealidad de la región de separación.

En la cuadro 2.1 se muestra el accuracy promedio (porcentaje de observaciones bien clasificadas) para las tres implementaciones la etiqueta 'kmeans' corresponde a la implementación del kernel base de R, la etiqueta 'kkmeans' corresponde a la implementación en el package 'Kernlab' y la etiqueta 'Kernel.kmeans' corresponde a mi implementación.
La figura 2.2 muestra las clasificaciones finales de los tres métodos.

Implentación	Accuracy
kmeans	.49
kkmeans	1
Kernel.kmeans	.72

Cuadro 2.1: El accuracy reportado por los dos últimos métodos utiliza un kernel gaussiano con $\sigma = \sqrt{0{,}5/2}$, para el caso de 'Kernel.kmeans' este fue el mejor accuracy obtenido con número de iteraciones $t = 20$ y un shift de -1.

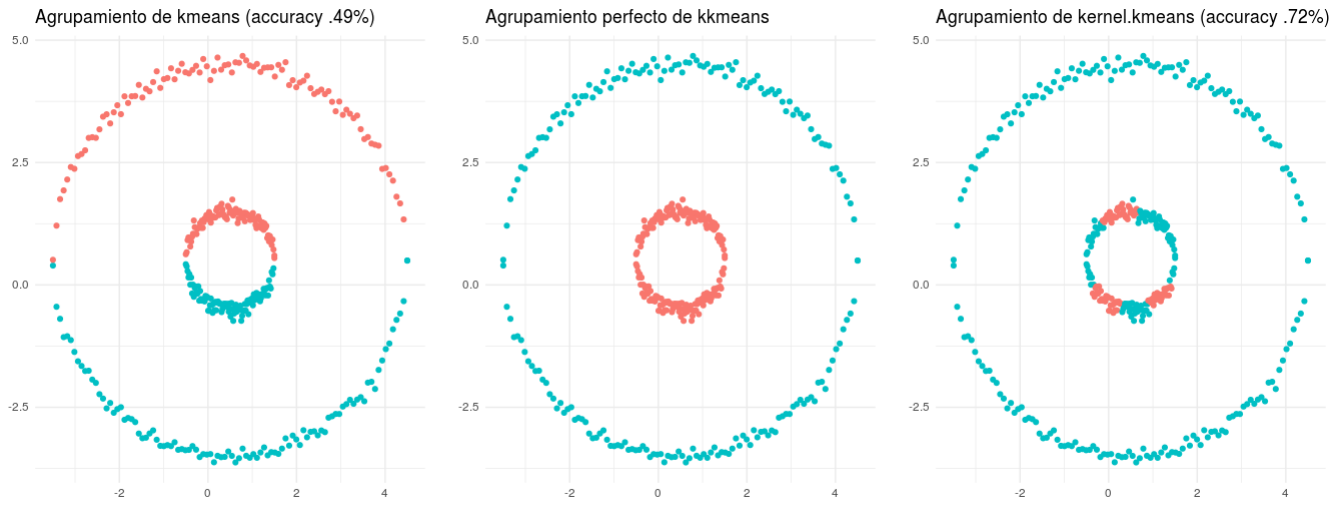

Figura 2.2: Clasificación de los puntos simulados promedio efectuada por 'kmeans' (izquierda), clasificación de 'kkmeans' (centro) y clasificación de 'Kernel.kmeans' (derecha).

Hasta este punto es importante señalar que la literatura dice que es de esperarse que 'kmeans' tenga ese pobre desempeño, en vista de la no linealidad en el conjunto de separación de los resultados. Un punto importante a destacar en esta sección es que los tiempos de ejecución entre la implementación 'Kernel.kmeans' y 'kkmeans' son similares aun sin utilizar el procesamiento multicore de 'Kernel.kmeans' (lo que aunque limita parcialmente la implementación ya que para utilizar más de un core se requiere usarla en un ambiente no-Windows)[1] y es importante destacar que las tres implementaciones se efectuaron con una **asignación de cluster inicial aleatoria de las observaciones**.
El siguiente conjunto de datos prueba que se considero es el que se utilizó en la tarea 1 de esta misma asignatura correspondientes a mediciones realizadas a 6497 muestras de vinos (rojos y blancos), donde nos interesa encontrar grupos según la calidad. En la tarea 1 (ejercicio 1) encontré que realizar un agrupamiento de la variable que mide la calidad y una rotación adecuada (obtenida con el software ggobi)[2] permite separar el conjunto de datos vinos por la calidad que presentan lo que induce a pensar que existe una frontera de separación entre los niveles de calidad de los vinos que es lineal. Sin embargo con la implementación actual la actual implementación tarda 5 hrs usando 6 cores en un ambiente

[1] Pues en el sistema operativo Windows, solo se puede usar un core de todos los disponibles en el equipo de computo, la implementación 'Kernel.kmeans' utiliza el package 'Parallel' para mejorar los tiempos de ejecución lo cual es útil si se dispone de más de un core. Sin embargo la implementación se puede utilizar en Windows pero con mayor tiempo de ejecución.

[2] Que después de leer el paper 'Computational Methods for High-Dimensional Rotations in Data Visualization' de Andreas Buja, Dianne Cook , Daniel Asimov y Catherine Hurley, sigo sin poder recrear

Linux por lo que se tomó una muestra aleatoria del 10 % del conjunto de datos de vinos, lo que generó un conjunto de datos de 650 observaciones.
En la cuadro 2.2 se muestra el accuracy promedio para las tres implementaciones considerando las etiquetas del cuadro 2.1 para el subconjunto de datos de vino.

Implentación	Accuracy
kmeans	0.12
kkmeans	0.12
Kernel.kmeans	0.16

Cuadro 2.2: El accuracy reportado por los dos últimos métodos utiliza un kernel gaussiano con $\sigma = 0{,}0013984457316839$, para el caso de 'Kernel.kmeans' este valor fue el obtenido con el valor default efectuado por 'kkmeans', con número de iteraciones $t = 20$ y un shift de -1.

En este punto es importante destacar la superioridad en tiempos de ejecución de 'kkmeans' sobre 'Kernel.kmeans' pues mientras la primera toma minutos de evaluación (debido a la forma en que calcula la matriz de Kernel que según su documentación consiste en utilizar la desigualdad del triángulo para evitar el cálculo de distancias innecesarias, de hecho su código fuente está construido por código R nato y no usa paralelización [3]) mientras que evaluar sólo la matriz de Kernel con 'Kernel.kmeans' en el conjunto de datos de vino completo requiere de 5 hrs usando 6 cores en paralelo en un ambiente Linux[4].
La figura 2.2 muestra las clasificaciones finales de los tres métodos, proyectadas usando PCA sobre la muestra con matriz de correlaciones

[3] Hasta donde me pude percatar de ello

[4] Al esperar durante esas 5hrs me percate de como mejorar la implementación haciendo uso de la tabla de indices que mencione al inicio de este ejercicio, pero esas mejoras las dejo para otra ocación, adémas solo reducirian a la mitad el tiempo de ejecución que no es competencia a los minutos que emplea 'kkmeans'

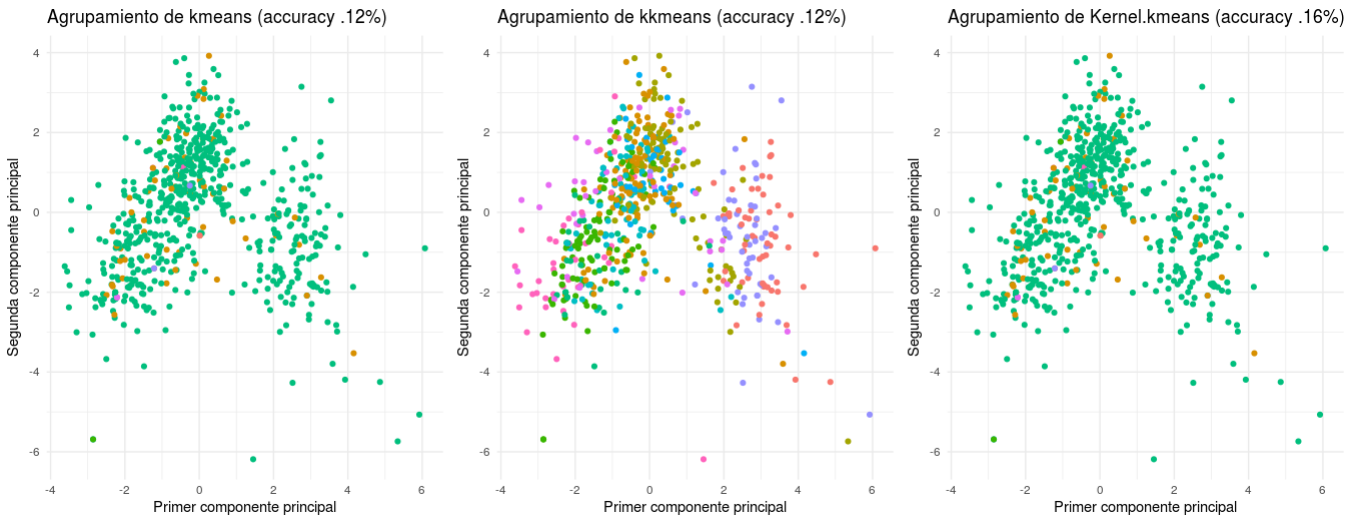

Figura 2.3: Clasificación de la muestra de vinos efectuada por 'kmeans' (izquierda), clasificación de 'kkmeans' (centro) y clasificación de 'Kernel.kmeans' (derecha).

A manera de conclusión y basándonos en las figura 2.3: aunque la implementación realizada destaca en su accuracy sobre las otras dos vemos que sus resultados están sesgados al igual que 'kmeans' pues etiqueta a la mayoría de las observaciones en la misma categoría de calidad (verde) mientras que 'kkmeans' tiene el mismo accuracy que 'keams' pero con una mayor diversificación de las observaciones en grupos.
Como conclusión de este ejercicio reiteramos que kernel k-means es mejor sobre conjuntos de datos que no son linealmente separables en su espacio original, en comparación con k-means; sin embargo (ambas implementaciones de Kernel k means) los resultados obtenidos no están por debajo de los de k-means en espacios linealmente separables [5] por lo que en ese sentido Kernel k-means es una generalización apropiada de k-means (que parece ciego a no linealidades). Sin embargo, como todo en la vida, presenta el trade-off de ser más general, pero requiere de optimizar hiperparámetros como σ para el kernel gaussiano y el shift, por solo nombrar algunos.

3. Ejercicio 3

Los datos en el archivo ***data_fruits_tarea.zip*** *contienen imágenes preprocesadas de* 100×100 *pixeles, que corresponden a diferentes tipos de frutas, tomadas en diferentes orientaciones y con diferentes características de forma y maduración. Supón que a una cadena de supermercados le interesa implementar un método automático de reconocimiento del tipo de fruta (y posiblemente su nivel de maduración) a través de las imágenes en color..*

[5] Como lo suponemos que es el espacio generado por la muestra del conjunto de datos de vinos

Para esta sección de la tarea se trabajó con el package 'imager' del entorno R. Se noto que la representación que este package hace de las imágenes es de arreglos tridimensionales donde las dos primeras dimensiones hacen referencia a la posición del pixel y la tercera dimensión a su valor en cada canal. La intensidad de color en cada canal, el valor del arreglo en su tercera dimensión sin embargo se mueve en un rango $[0, 1]$ discreto en contrapunto de la representación de valores discretos en $[0, 255]$ como se esperaba, lo que abrió la posibilidad de que el análisis realizado dependiese de la representación por parte del package, para verificar la certeza de este punto se realizó el análisis correspondiente a los dos primeros puntos con la representación natural de las imágenes que realiza el package, posteriormente se realizó el mismo análisis pero multiplicando los arreglos tridimensionales por 255 (obteniendo la representación que se esperaba en un inicio) y no se encontraron diferencias, por lo que esta verificado que el escalamiento no altera los análisis siguientes y se trabajo con los valores discretos en $[0, 1]$.
En un primer acercamiento con las imágenes con las que se trabaja en este ejercicio, es decir al abrirlas y verlas en secuencia se notó que el preprocesamiento segmentó las siluetas de las frutas sin embargo como las imágenes estan estandarizadas a un tamaño de 100×100 al espacio entre los pixeles que conforman la fruta y el tamaño fijo se relleno con pixeles en color blanco; en un primer análisis se realizó una exploración con una visualización en 3D (en el espacio RGB) de las frutas representadas por su mediana en cada canal, posteriormente se realizo PCA sobre la matriz de correlaciones de los datos obtenidos para realizar el inciso 2 de este ejercicio (ver las frutas proyectadas en sus dos primeras componentes) y se obtuvieron resultados prudentes es decir que en la visualización 3D se notan grupos de frutas (es importante mencionar que en esta visualización el grupo que forman las imágenes de las fresas y del aguacate se mueven en rectas hacia el color blanco, el punto situado en la coordenada $(1, 1, 1)^t$ lo que atribuimos a que todos los pixeles blancos que forman el fondo, en estos casos las imágenes eran más pequeñas que en los otros conjuntos de imágenes, logran sesgar la distribución de los valores en cada canal moviendo la mediana) sin embargo en la proyección en las dos primeras componentes es fácil notar que las proyecciones de las frutas en este espacio hace que algunos grupos se traslapen por lo que sería difícil clasificar utilizando solamente esa información (pese a que las dos primeras componentes explican poco más del X % de varianza total) si no tuviéramos la etiqueta que identifica al tipo de fruta en la imagen.
Por lo que se decidió realizar una etapa adicional de preprocesamiento a las imágenes, la cual consiste en calcular las medianas que se requieren pero solo sobre los pixeles que distan menos de 0.02037707 del pixel blanco, con coordenas $(1, 1, 1)^t$, en la métrica L_2. Con lo anterior se busca disminuir la cantidad solamente de pixeles blancos en los tres canales (a diferencia de lo que representaría truncar los valores cercanos a 1 en cada canal lo cual se comprobó que induce ruido poco provechoso en el análisis) con la finalidad de que estos pixeles sesguen en menor medida la distribución de los valores en cada canal y las estimaciones de las medianas sean más apropiadas. En los tres incisos siguientes se reportan los resultados sobre las imágenes después de esta etapa de preprocesamiento.

1. *Obten una representación de las imágenes en el espacio **RGB** usando la mediana como medida de resumen de los valores en cada canal ¿Puedes identicar patrones interesantes en esta representación?*

   Al visualizar en 3D, en el espacio (r,g,b), el conjunto de datos de las imágenes representadas por su mediana en cada canal, obtenemos la visualización que se muestra en la figura 3.1 donde podemos apreciar que las imágenes de la fruta 'Carambula' se mezclan con las de la fruta 'Apple_Golden' sin embargo la identificación de grupos se vuelve más difícil alrededor del punto (0.4,0.3, 0.1) pues en esa vecindad los puntos de los grupos correspondientes a las frutas 'Strawberry', 'Apple_braburn', y 'Peach' se sobreponen haciendo difícil su identificación sin las etiquetas que se conocen. También es importante notar que los puntos correspondientes a los grupos de frutas 'Apricot' y 'Pineapple' se mezclan en menor medida que el conjunto de frutas mencionado anteriormente. Sin embargo, el grupo formado por las frutas 'Huckleberry', 'Cherry' y 'Avocato' se distinguen sencillamente de los demas puntos.

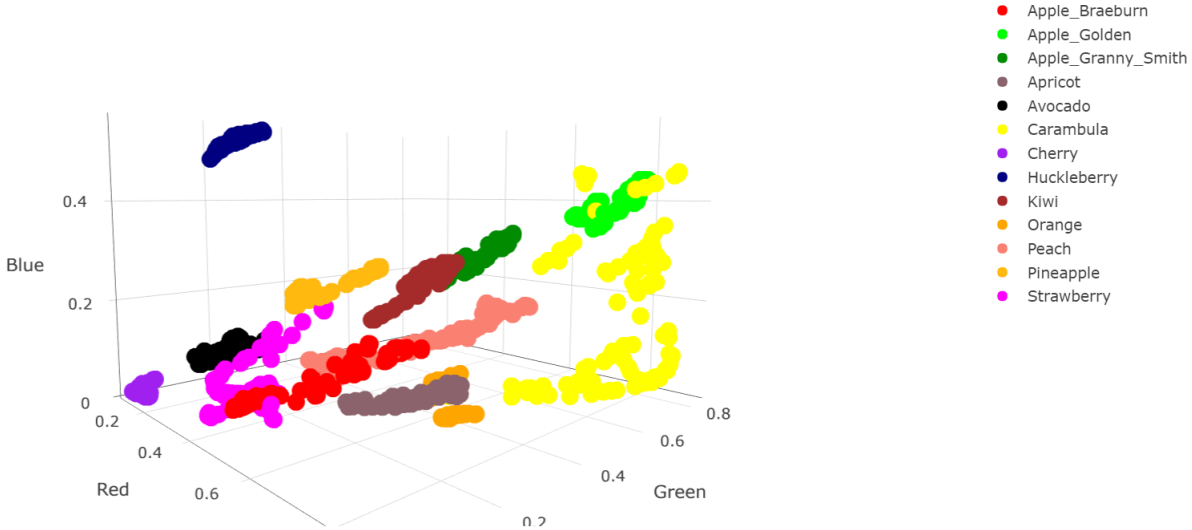

Figura 3.1: Representación de las imágenes por sus medianas en cada canal en el espacio (r,g,b).

2. *Realiza PCA y Kernel PCA con un kernel Gaussiano en los datos que obtuviste. ¿Puedes identicar grupos interesantes o informativos de las imágenes en los primeros componentes principales?*

   Después de realizar PCA, usando la matriz de covarianzas, y kernel PCA sobre el

conjunto de medianas de las imágenes, y proyectar en las dos primeras componentes obtenemos que las dos primeras componentes principales de PCA explican el 96.73 % de la varianza total mientras que al efectuar kernel PCA con un kernel gaussiano, con parámetro $\sigma = 1/90$ se obtiene que las dos primeras componentes explican el 100 % de la varianza total. En ambas proyecciones, como podemos ver en la figura 3.2 se distinguen (por estar separados) los conjuntos de frutas marcados como 'Huckleberry', 'Avocato', 'Orange', 'Apple_Granny_Smith' y 'Kiwi'. En contraposición a la imagen 3.1 en las proyecciones de la figura 3.2 es más sencillo notar que los conjuntos formados por los puntos de las frutas marcadas como 'Strawberry', 'Peach' y 'Apple_Breaburn' se sobreponen, más aún ahora el conjunto formado por los puntos de la fruta 'Carambula' se mezclan con los de las frutas 'Pineapple', 'Apricot' y nuevamente con 'Apple_Golden'.

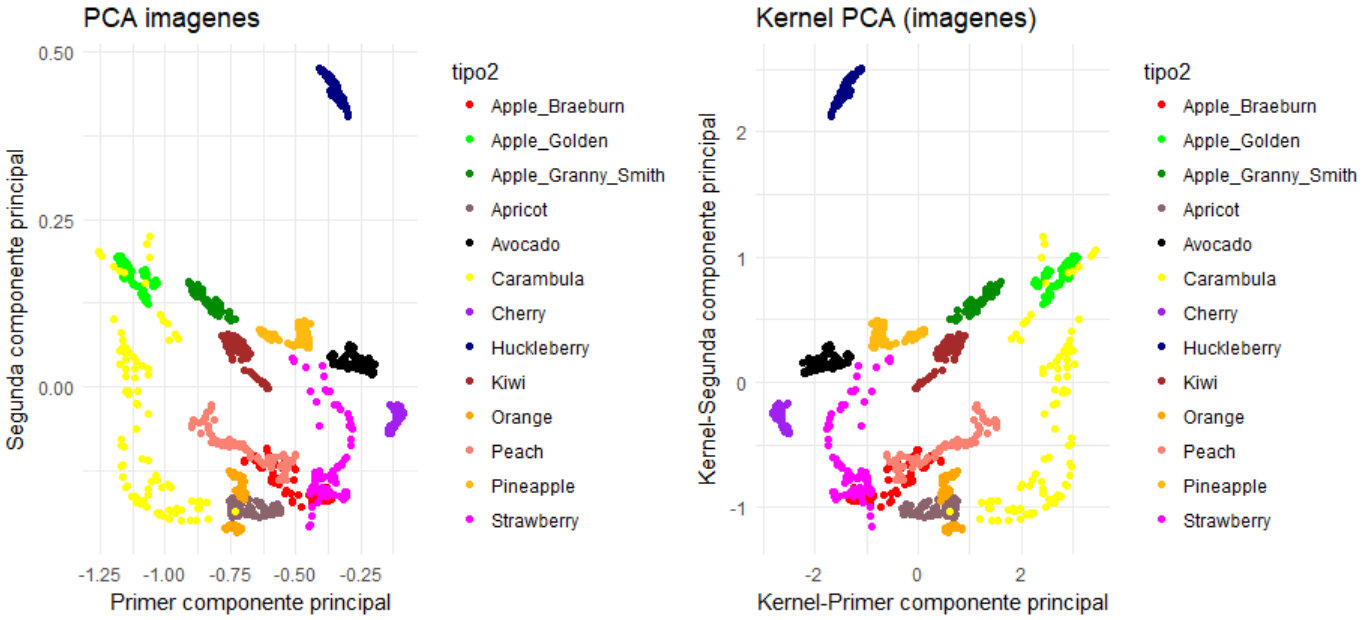

Figura 3.2: Proyección en las dos primeras componentes principales, usando PCA con matriz de covarianzas (izquierda) y usando kernel PCA con kernel gaussiano y $\sigma = 1/90$ (derecha). Ambas proyecciones son igual de informativas, aunque difieren en su dirección.

3. *Aplica K-means y Kernel K-means. Verica si puedes identicar los diferentes grupos de frutas.*

Después de aplicar k-means sobre el conjunto de datos se logró un error de clasi-

ficación de 24.6 % en contraste utilizando kernel k-means con un kernel gaussiano la mejor clasificación que se obtuvo tiene un error de 29.8 %, pese a que se utilizaron otros kernels la diferencia en cuanto al error de clasificación no es significativa. En la figura 3.3 se muestran las clasificaciones de los algoritmos en el conjunto de datos proyectado en las dos primeras componentes principales de PCA utilizando la matriz de covarianzas. Por lo aprendido en los ejercicios anteriores, concluimos que el algoritmo de kernel kmeans y kernel PCA no logran contrastar la distribución del color en las imágenes debido a que la mediana de cada canal no contiene información suficiente para ello.

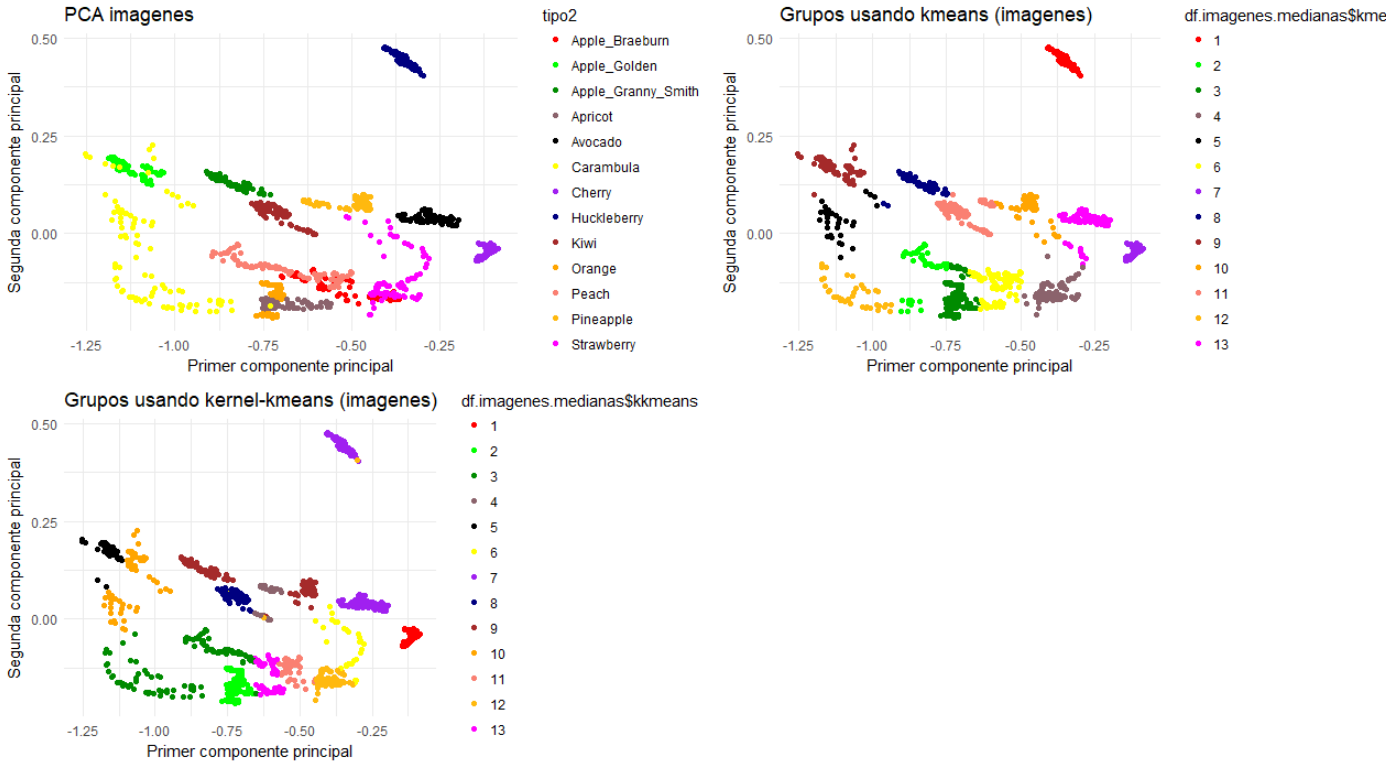

Figura 3.3: Resultados de clasificación de las imágenes usando el hecho de que a priori se conoce que son 13 grupos proyectado en las dos primeras componentes principales usando PCA con matriz de covarianza, la clasificación conocida (izquierda-arriba), la mejor clasificación obtenida con kmeans (derecha-arriba) y la mejor clasificación obtenida con kernel kmeans usando un kernel gaussiano (izquierda-abajo).

4. *Repite los incisos anteriores usando el espacio* ***HSV*** *(Hue, Saturation, Value). Pa-*

ra incluir más información sobre cada dimensión, utiliza la información de los tres cuartiles centrales en cada una de ellas, de forma que tengas una representacion en un espacio de tamaño $d = 9$. ¿Notas alguna mejoría?

Al realizar el analisis que se hizo en los tres últimos incisos (sobre el conjunto de imágenes tal y como se nos proporcionaron) el desempeño de kernel-kmeans y de kmeans era pobre. En un segundo intento se agrego más información, trabaje con los deciles sin obtener mejoras en el desempeño de los algoritmos de cluster (kmeans y kernel kmenas). Después de visualizar la información con la que trabajaba (en conjuntos de 3, por ejemplo grafique el primer decil en cada canal) me percate de la poca variabilidad que presentaban las variables y lo dificil que sería identificar grupos de esta forma. En particular el primer decil (aunque no lo documento apropiadamente con una gráfica) de los canales 'H', 'S' y 'V' era un contraejemplo de una variable informativa pues todos los puntos estaban alineados en una recta vertical pasando por el cero. En vista de los resultados anteriores tome una muestra de 50 imágenes y me di a la tarea de ver su descomposición en los canales H,S y V por separado. Note que mientras unas imágenes tenían "mucha" información en el canal H, en el canal S se tenia poca información y casi nada en el canal V, por otro lado, algunas imágenes tenían mucha información en el canal S, poca en el canal H y casi nada en V, y en un reducido número de imágenes de mi muestra algunas imágenes poseían mucha información en el canal V y casi nada en los otros dos. Por lo anterior deduje que sí sería posible distinguir con la información del espacio HSV pero algo estaba sesgando los resultados, así que realice de nuevo el proceso que menciono antes del inciso 1) de este ejercicio el cual consiste en quitar los pixeles muy cercanos al pixel blanco (la métrica que uso es la L_2 y elimino piexeles que distan menos de 3 pixeles con respecto al blanco) y me di a la tarea de visualizar uno por uno los resultados. Note que el efecto del preprocesamiento era más evidente en el canal H generando imágenes 'más informativas' o con menos ruido. Concluyo que los pixeles de sobra que ya he mencionado que rellenan las imágenes para estandarizar su tamaño a 1000×1000 afectaban sobre todo a la transformación no lineal del espacio RGB a HSV. En lo que sigue reporto los hallazgos sobre las imágenes preprocesadas como lo hice anteriormente y de ahí a convertirlas a HSV y ahí hacer las mediciones de los cuartiles requeridos.
En lo que respecta al punto inicial, al graficar el primer y tercer cuartil en un espacio euclidiano los puntos que representan a las imágenes estan más aglomerados que los puntos que tenemos en la figura 3.1, sin embargo, de nuevo la mediana parece ser más informativa en este espacio como lo podemos ver en la figura 3.4 los imágenes representadas por sus medianas en los canales HSV estan más aglomerados en comparación de su representación en RGB.

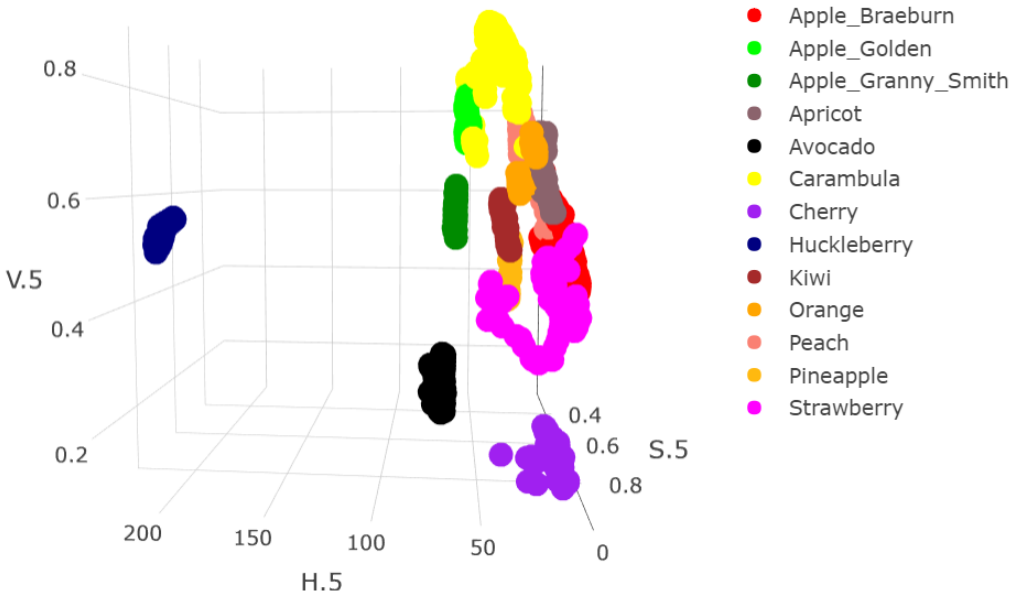

Figura 3.4: Representación de las imágenes por sus medianas en cada canal del espacio HSV, la grafica considera coordenadas en un espacio R^3 con la métrica natural del espacio, es decir la visualización no considera las coordenadas del espacio HSB.

Posteriormente se realizo PCA (utilizando la matriz de covarianza) sobre el conjunto de mediciones que se tiene de las imágenes, es decir sobre las variables que contienen el cuartil 0,25, 0,5 y 0,75, las dos primeras componentes principales contienen más del 99 % de la varianza total. Por su parte al efectuar kernel PCA sobre el mismo conjunto de datos, y tras evaluar diferentes kernels, se elegió el kernel gaussiano con un parámetro $\sigma = ,004$ como el que contiene más información en sus dos primeras componentes a la par de efectuar una proyección en estas componentes que hiciera sencillo identificar a los grupos de imágenes, las dos primeras componentes principales de kernel PCA explican solamente poco más del 35 % de la varianza total. En la figura 3.5 podemos ver los resultados obtenidos, mientras que la proyección de PCA captura más varianza en ese espacio es muy dificil distinguir grupos de frutas de hecho se aglomeran la mayoría de los puntos alrededor del origen, en esta proyección solo es sencillo distinguir al grupo de la fruta 'Huckleberry' (lo cual desde la imagen 3.4 era fácil notar), 'Cherry' y en menor medida es fácil identificar de manera aislada a los grupos de las frutas 'Apple_Golden', 'Avocato' y 'Apple_Granny_Smith' mientras que las frutas 'Carambula', 'Orange', 'Kiwi', 'Pineapple', Peach' y 'Apricot' se sobreponen en la vecindad del origen. En contraste la proyección en las dos primeras componentes

principales de Kernel PCA, como ya mencionamos con un kernel gaussiano, es no lineal y permite identificar fácilmente los mismos grupos que PCA aunado a que la fruta 'Carambula' es más fácil de aislar, los grupos de frutas que ya mencionamos que se sobreponen en la proyección de PCA (con excepción de la 'Carambula') se siguen sobreponiendo sin embargo en esta proyección el traslape entre clases se da a pares con excepción de los grupos de 'Peach', 'Apple_Braeburn' y 'Apricot' que se enciman en algunos puntos. En general esta proyección es más informativa que la de PCA.

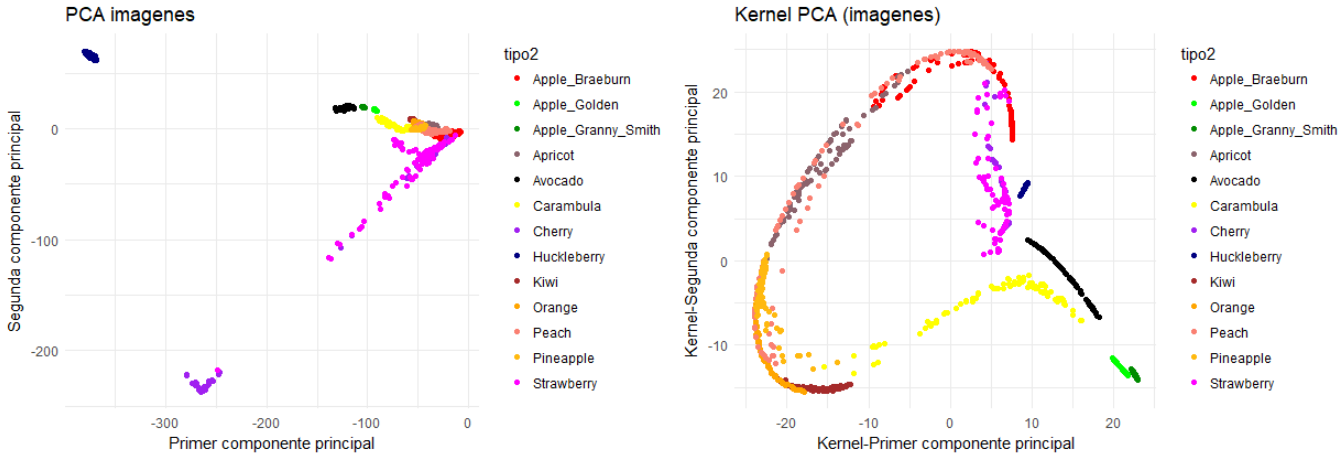

Figura 3.5: Proyección de los datos en las dos primeras componentes principales. Usando PCA con matriz de covarianza (izquierda) y usando un kernel gaussiano con $\sigma = ,004$ (derecha).

Finalmente se efectuó clustering utilizando Kmeans y Kernel kmeans, en el primer caso el mejor desempeño logrado (conociendo de manera a priori que se cuenta con 13 grupos) tiene un error de poco más del 30 % mientras que, en el segundo caso, utilizando un kernel gaussiano con el mismo valor del parámetro que en el párrafo anterior, el mejor desempeño tiene un error de clasificación de poco mas del 30 %. En la figura 3.6 se decidio mostrar los resultados de las clasificaciones de los dos métodos de clustering en el espacio de las dos primeras componentes principales de PCA, aunque no es lo indicado para el método de Kernel kmeans en vista de que este método agrupa con una similaridad diferente a la métrica euclidiana que utiliza kmeans, debido a que en este espacio es fácil ver el clásico agrupamiento efectuado por kmeans, donde podemos ver que debido a la cercanía considerando la métrica L_2 entre los puntos de las frutas 'Peach' y 'strawberry' son etiquetados como iguales por kmeans además este algoritmo divide los puntos de 'Strawberry' hasta en tres grupos diferentes. Por su parte los resultados de usar kernel kmeans no divide al grupo de puntos de 'Strawberry' sin embargo lo confunde con el de 'Cherry' y al igual que

kmeans el grupo de puntos correspondientes a 'Huckleberry' es etiquetado por más de un grupo.

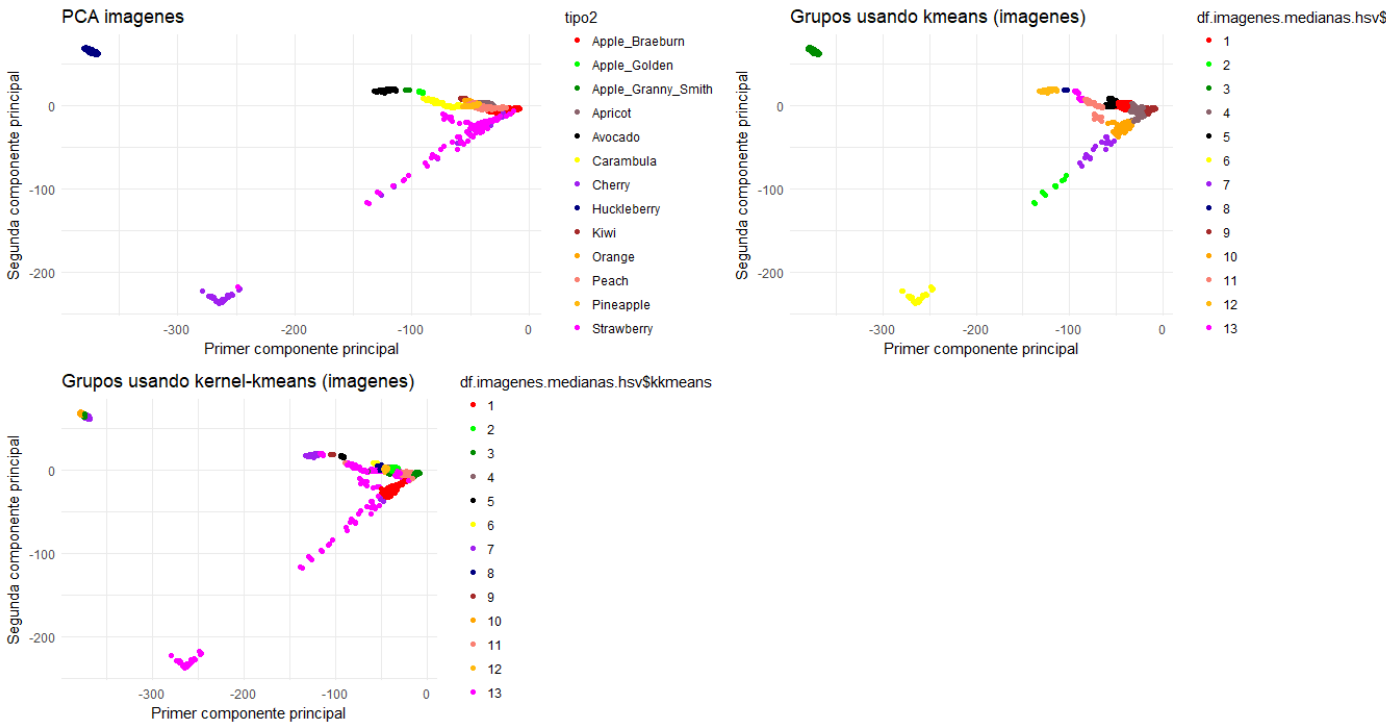

Figura 3.6: Resultados de clasificación de las imágenes representadas cada una por sus cuartiles 0.25, 0.5 y 0.75 en los canales del espacio HSV. Conociendo que son 13 grupos, los resultados se muestran en las dos primeras componentes principales de PCA con matriz de covarianza, la clasificación conocida (izquierda-arriba), la mejor clasificación usando kmeans (arriba-derecha) y la mejor clasificación usando kernel kmeans con un kernel gaussiano y $\sigma = 0{,}004$ (izquierda-abajo

Concluimos este ejercicio con varias reflexiones, primero como notamos en el inciso 1) y 4) la identificación de los grupos parecía una tarea sencilla pues incluso en el espacio HSV si bien los grupos no son linealmente separables sí se pueden acotar cada uno, lo cual es fácil de ver en la figura 3.1 y la tarea se vuelve más compleja en el espacio HSV como lo ejemplifica la figura 3.4. Si bien nuestra intuición euclidiana (pues estamos acostumbrados a percibir en un espacio euclidiano de dimensión 3 y solemos utilizar la norma L_2 para medir distancias) nos permite identificar patrones en el espacio euclidiano (R, G, B) es natural que kmeans encuentre de mejor manera los grupos 'correctos' para las imágenes considerando su representación en RGB. Segundo, en contraposición a lo que solemos pensar que un algoritmo más

complejo tendrá mejores resultados pudimos ver en el inciso 2) y 4) que no es verdad ya que el cluster que utiliza kernel tiene un mayor error que kmeans incluso en el espacio no euclidiano HSV. Además, utilizar un método de clustering que considera kernels hace más laborioso su uso (pues hay hiperparámetros que optimizar en cada tipo de kernel) además de que la interpretabilidad de los grupos que realice es más complicada porque son clusters que se construyen en un espacio de dimensión infinita.

Tercero y último, el método de kernel kmeans contrasta las densidades de los clusters que construye. Entonces la reducción de dimensión que realizamos de las imágenes a solo medianas o cuartiles parece indicar que no son lo suficientemente informativas para que este método destaque en desempeño. Si bien como ya mencione en párrafos anteriores consideré usar todos los deciles por canal de HSV (sin embargo esa prueba la realice con los datos sin el preprocesamiento que efectuó) sin tener mejoras en el desempeño de kernel kmeans por lo que es importante prestar atención a dos cosas: primero que kernel kmeans no logro contrastar las densidades de los grupos debido a la poca información que tiene de los grupos (solo usamos medianas y cuartiles) o bien la buena estandarización de las imágenes no captura esta variabilidad pues como consecuencia del preprocesamiento que realizó y de la verificación que hice para una muestra de la descomposición en canales de HSV pude percatarme de que las imágenes fueron tomadas de manera continua y con las mismas condiciones ambientales (lo cual en la practica es muy dificil de tener) y de luz (inclusive pude detectar en que dirección estaba la fuente de iluminación de las frutas) así pues hay varios factores que interfirieron en el bajo desempeño de kernel-kmeans.
El código correspondiente a este ejercicio lo agrego en el anexo C, así como en el archivo llamado 'ejercicio3.r'

4. Ejercicio 4

*Este ejercicio es sobre análisis de sentimientos. En el archivo m**ovie_reviews.zip** se encuentran las críticas de 500 películas tomadas de `http://www.imdb.com1`. En **movie_reviews.csv** se encuentra la información de cada película, incluyendo, entre otra cosas, su nombre, el nombre del archivo de texto donde se encuentra su crítica y el **sentimiento** relacionado con la misma: P (positiva) o N (negativa).*

En un primer intento intente realizar PCA sobre la matriz de términos documentos considerando todas las palabras, o que generaba una matriz de correlación o de covarianzas de 22260×22260 lo que no hizo factible seguir esta idea. Como parte del preprocesamiento en este primer intento considere no remover los números ni los signos de puntuación, los resultados fueron que los números presentes en ambos conjuntos de datos son del tipo '19990', '20002' (suponemos que hacen referencia al año de proyección de la película) en contra punto se esperaban números en el rango discreto $[0, 10]$ haciendo alusión a la calificación

que la persona que realiza la reseña da a la película. En cuanto a los signos de puntuación se consideraron incluir pues símbolos de emoticones como ':D' y ':(' pudieran da información, sin embargo en los documentos no se encontraron signos de puntuación interesantes por lo que se descarto la idea de usarlos.

El proceso de preprocesamiento de los textos que seguí consiste en remover espacios en blanco, remover números, trabajar solo con palabras en minúsculas, remover las stopwords del idioma ingles y realizar la extracción de raíces con el algoritmo de Porter. Considero todas las palabras independientemente de su frecuencia en el documento.

1. *Realiza PCA en la representación de los textos obtenida con* ***bag of words*** *usando los n términos más frecuentes (decide el valor de n). Realiza el preproceso que consideres necesario en los textos. ¿Puedes identicar las críticas positivas y negativas en los componentes principales? Si la respuesta es no, explica las razones.*

   Como parte del preprocesamiento para saber que palabras considerar para construir la matriz que requiere PCA, aproveche que el conjunto de datos es pequeño y de mi a la tarea de explorarlo. Primero obtuve todas las palabras y sus frecuencias en el corpus formado por los documentos marcados como reseñas positivas he hice lo mismo para el corpus formado por los documentos de todas las reseñas marcadas como negativas. Obtuve una lista de palabras en común (con un simple join) en la figura 4.1 se muestran puntos que representan este conjunto de palabras en común los ejes señalan la frecuencia de la palabra en el corpus de positivos o negativos, lo que sugiere una correlación entre las frecuencias de este conjunto de palabras el cual descarté del corpus 'completo' (el que se forma con todas las reseñas). Además de figura 4.1 realice una prueba de correlación de Pearson que arroja un p-value cercano a cero de hecho es menor a 2.2e-16, por lo que podemos suponer que estas palabras no agregan información para nuestro propósito.
   Posteriormente filtre de la matriz de términos-documentos del corpus 'completo' las filas que contienen a las palabras de la lista de palabras 'comunes' que detecte y que no son informativas (la lista de la que hablo en el párrafo anterior). La lista de palabras comunes tiene una cardinalidad de 9477. Posteriormente después de varios intentos determine que usaría solo las palabras restantes con frecuencia en el corpus completo mayor a 5, así que nuevamente filtre la matriz de términos documentos con el criterio de que las palabras tuviesen además una frecuencia mayor a 5. Por lo que finalmente la matriz de términos documentos que empleare para hacer PCA tiene las dimensiones de 443×1000. Así que el n por el que pregunta el ejercicio en mi caso es de 443, sin embargo, no es exactamente el número de términos más frecuentes pues algunos de ellos se perdieron al sacar del analisis las palabras en común de ambos corpus, como podemos ver en la figura 4.1 no considere términos que tienen frecuencia de hasta 2500 tan solo en el corpus de los documentos 'negativos'.

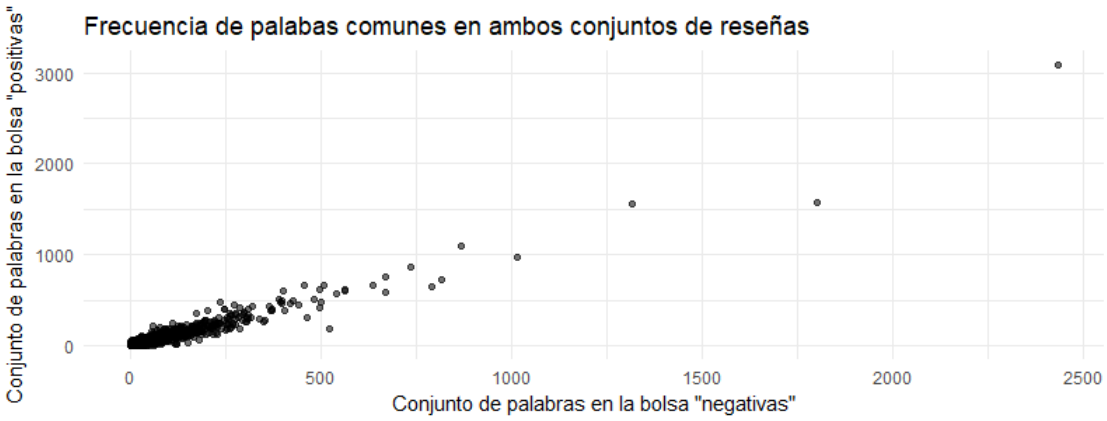

Figura 4.1: Diagrama de dispersión de las palabras en común de los corpus 'positivo' y 'negativo' nótese que existe correlación positiva, este conjunto de palabras se descargo en el analisis, pues se consideraron como palabras poco informativas.

Luego realice PCA sobre la matriz de correlación de la matriz de términos-documentos, pese a que las dos primeras componentes principales solo explican poco más del 2.8 % se obtuvo un resultado prudente en la clasificación usando las primeras dos componentes. En la figura 4.2 se muestran las proyecciones de los conjuntos de documentos en las dos primeras componentes principales, como podemos ver el criterio de que **si el documento es positivo en la primer componente principal** (después de rotar apropiadamente el data.frame que contiene la información de las frecuencias de los términos en los documentos) y **si es negativo en la segunda componente principal** entonces decimos que es negativo en caso contrario la reseña es positiva, es decir que las dos primeras componentes principales se pueden interpretar como el sentimiento de la persona (su personalidad, su forma de escribir) y la otra como su opinion respecto a una película en concreto, ambos aspectos estan relacionados, pero pueden ser independientes en la practica ya que alguien con tintes pesimistas tambien puede dar una reseña positiva y viceversa. Con el criterio antes mencionado se tiene un error de clasificación de 34.4 %. Donde las criticas negativas se clasifican perfectamente mientras que las positiva no en su totalidad, de hecho 344 criticas positivas usando el criterio mencionado se clasifican como negativas

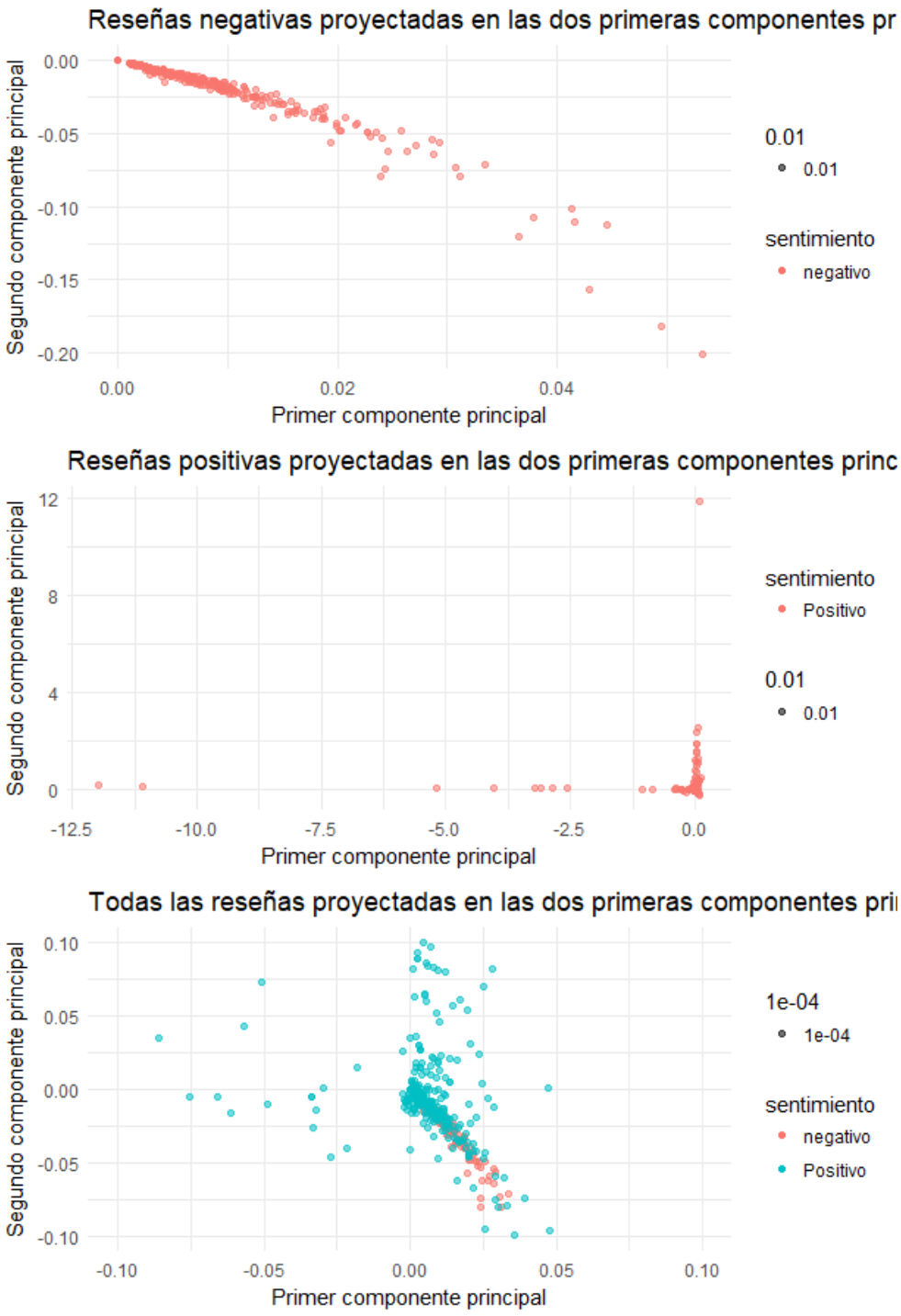

Figura 4.2: Proyección de las reseñas en las dos primeras componentes principales utilizando la matriz de correlación, arriba se muestra la proyección solo de las reseñas etiquetadas como negativas, en medio las reseñas etiquetadas como positivas y abajo se muestran todas las reseñas en conjunto.

2. *Redefine los datos de entrada de* ***bag of words*** *uniendo k documentos de cada categoría, por ejemplo* $k = 5$*. Repite el inciso anterior. ¿Qué diferencias notaste? Describe tus hallazgos.*

   De manera análoga al inciso anterior primero agrupé los documentos secuencialmente en grupos de 5, obtuve las palabras que considero comunes a ambos corpus, el de las reseñas positivas y el de las reseñas negativas, encontré que el conjunto de palabras comunes es el mismo que en el caso anterior, lo cual no es de sorprender por el concepto mismo de 'bag of words' sin embargo fue un buen ejercicio comprobarlo.
   Posteriormente repliqué lo que hice en el primer inciso, para $k = 2$ y $k = 5$, sin embargo, con $k = 5$ se obtuvieron mejores resultados de clasificación, es una pena pues la posibilidad de $k = 2$ es menos restrictiva en términos prácticos.
   En la figura 4.3 se proyectan en los dos primeras componentes principales utilizando la matriz de correlación (que explican el 4.7 % de la varianza total) para nuestro corpus con documentos k-documentos, que son el resultados de concadenar los k documentos continuos, en nuestro caso repito $k = 5$ da buenos resultados de clasificación por lo que incrementar k es una cuestión que si bien podría mejorar el desempeño de la clasificación involucra en la practica tener documentos más de 5 veces de largo de lo que ya lo son las reseñas. Agrupando los documentos en conjuntos de 5 se logro una clasificación con un error de 2 % de los 5-documentos (lo que se podría interpretar en un 10 % de los 1-documentos).

   Como la figura 4.3 sugiere el criterio de clasificación consiste en asignar sentimiento negativo si el primer componente principal de la observación en este caso un 5-documento es negativa y tambien lo es el valor del 5-documento en su segunda componente. Nótese que la separación es más tangible que en el inciso anterior.

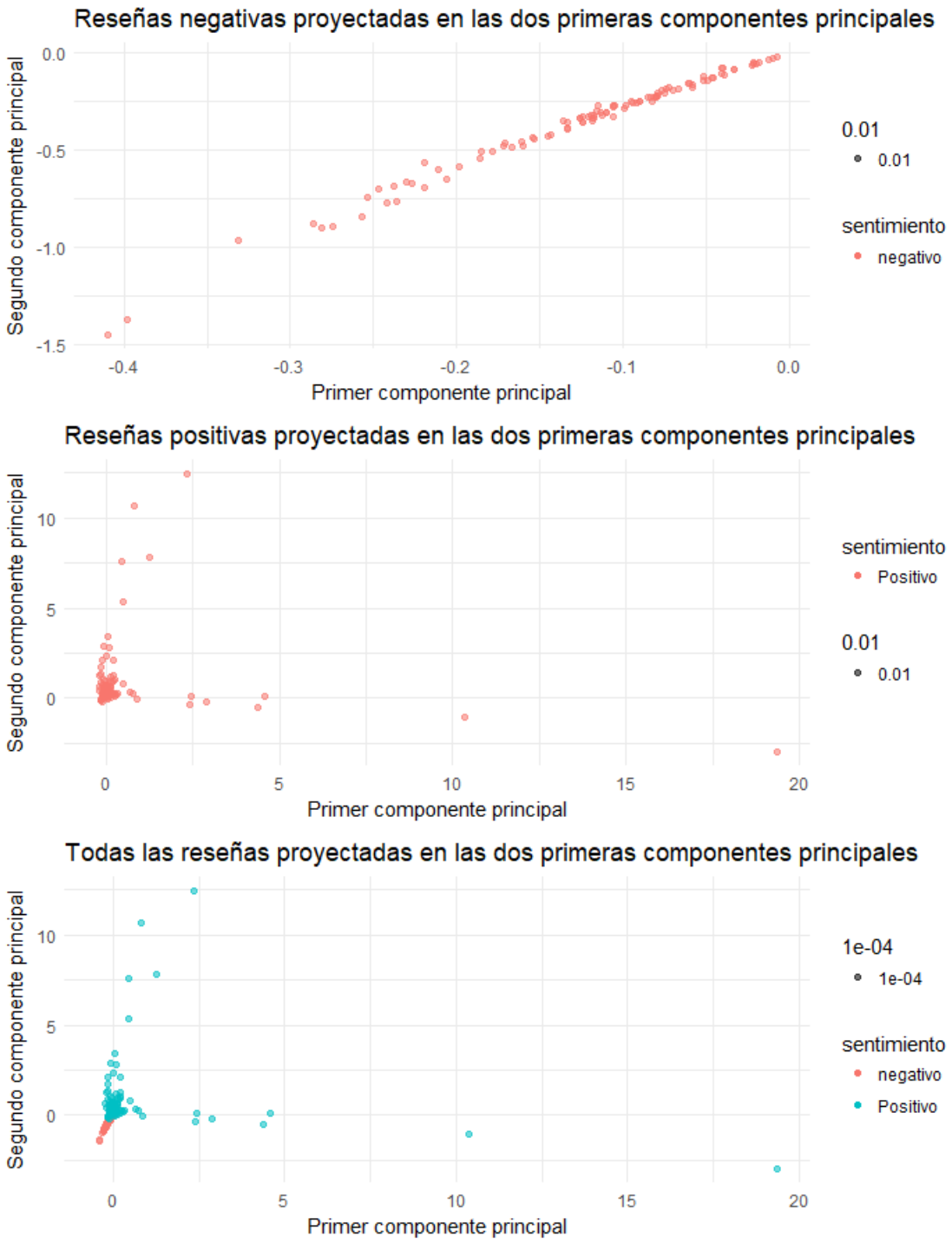

Figura 4.3: Proyección de las reseñas agrupadas en conjuntos de 5, lo que llamamos 5-documentos, en las dos primeras componentes principales utilizando matriz de correlación. Arriba se muestra la proyección con las reseñas etiquetadas como negativas, en el centro las marcadas como positivas y abajo se muestran ambas.

Por último, el código de este analisis se reporta en el apéndice D así como en el archivo llamado 'ejercicio4.r

5. Ejercicio 5

Este ejercicio es sobre análisis de similaridad en textos. El archivo ***train_stock.csv*** *contiene texto descriptivo de acciones en la bolsa. El texto contenido en las columnas* ***description_x*** *y* ***description_y****, se considera que se refieren a la misma acción si ambos tickers* ***ticker_x*** *y* ***ticker_y*** *coinciden, aunque las descripciones no sean iguales.*

1. *Realiza Kernel PCA con string kernels para analizar la similaridad de los textos* ***description_x*** *y* ***description_y****. Describe los patrones o grupos signicativos que logres identicar en los primeros componentes principales. Prueba con diferentes tipos de string kernels y reporta cuál te proporciona el "mejor" resultado.*

   Una observación que me resulto útil para saber que kernel y que parámetros (del mismo kernel) utilizar fue prestar atención en 'qué tan parecidos son los strings marcados como coincidentes' y que tan 'diferentes son entre sí los marcados como no coincidentes'. Para ello seleccione cada grupo, el primero de coincidentes seleccionando las observaciones con valor de 'TRUE' en la columna 'same_security' (cuya veracidad comprobé a pie y efectivamente refleja cuando el 'ticker_x' es igual a la columna 'ticker_y' en la observación), posteriormente me fijé en cuantas palabras contiene el string en el campo 'description_x' y cuantas palabras contiene el string en el campo 'description_y' de lo cual pude obtener que en esta submuestra las descripciones coincidentes tienen en su mayor proporción 3 y 4 palabras de longitud. Hice lo análogo para el subconjunto de observaciones marcadas como 'FALSE' en la columna 'same_security' y obtuve que la mayor proporción de descripciones no coincidentes tienen una longitud diferente en general, pero las longitudes 5 y 6 reportan mayor presencia.
   Con base en lo anterior comencé a utilizar el kernel string 'spectrum' con longitud variable y un parámetro λ también variable en el intervalo $[0, 1]$. Posteriormente utilice otros string kernel como el 'constant' (el cual parecía prometedor, pero en vista de la variabilidad de la longitud de las descripciones en las observaciones no coincidentes se explica el rendimiento poco menor respecto al mejor que reporto posteriormente) también el string kernel 'boundrange' parecía viable, pero se enfrenta al mismo reto que el 'constant'.
   En la figura 5.1 se encuentran las proyecciones en las dos primeras componentes de kernel-pca sobre el conjunto de datos con el que se trabajó, las observaciones estan etiquetadas respecto a la columna 'same_security', utilizando un kernel gaussiano con $\sigma = 1$ (pues cuidamos que el kernel string estuviese normalizado). También en la figura 5.1 se refleja que el kernel string que resulto 'más útil' pues permite identificar de manera menos complicada que los otros string kernels probados, es el 'spectrum'

con un valor de $n = 3$ para la longitud de las cadenas y un $\lambda = 1$. Es importante mencionar que durante la fase de exploración se encontró que los resultados de las proyecciones en las dos primeras componentes de kernel-pca son poco sensibles a cambios en el parámetro λ, manteniendo siempre el kernel de pca como un gaussiano con $\sigma = 1$, mientras que cambios en el método de string kernel sí eran notorios, en los casos en que el string kernel requiere de un parámetro n también este cambio se refleja ligeramente en las proyecciones.
Si bien los resultados no son idóneos, consideremos que el rango de la matriz de kernel string es mayor a 45 que esta normalizada y que solo podemos plasmar 2 componentes, en la figura 5.1 podemos apreciar que hay confusión en la distinción de las observaciones alrededor del origen sin embargo en la dirección positiva de la segunda componente destacan las observaciones marcadas como no coincidentes (en rosa) al igual que en la dirección negativa de la primer componente manteniendo positiva la segunda componente.

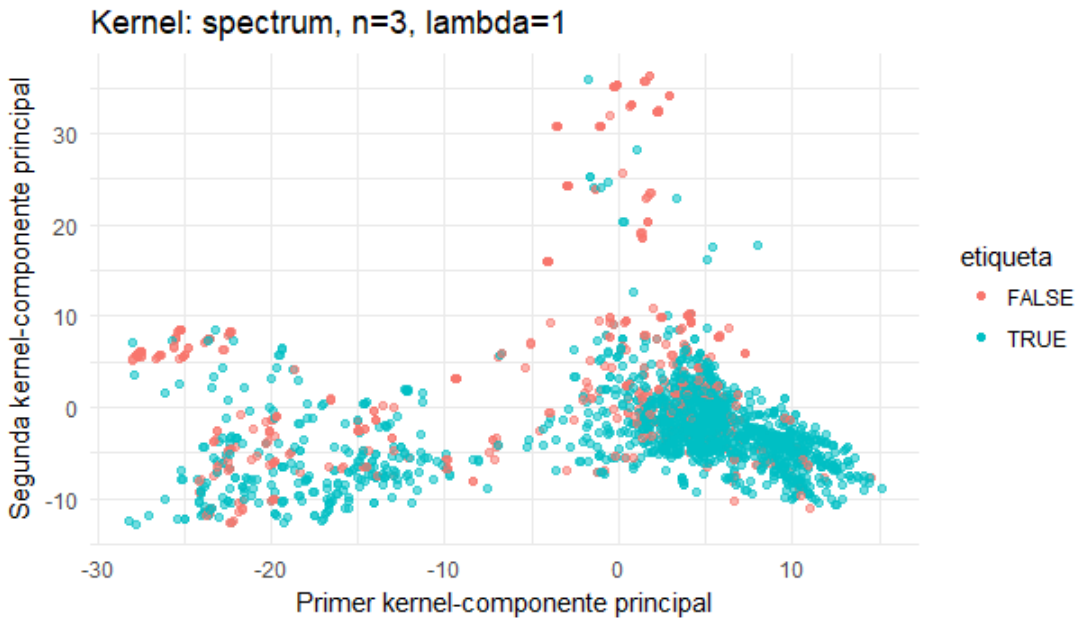

Figura 5.1: Proyección de los datos 'train_stock' en las dos primeras componentes principales de kernel-pca (con kernel gaussiano y $\sigma = 1$) utilizando un kernel string (para construir similaridades normalizadas) 'spectrum' que considera palabras de longitud 3 y que penaliza con un parámetro de $\lambda = 1$.

2. *Considera los textos de* ***description_x*** *como tu conjunto de entrenamiento y realiza Kernel PCA como en el inciso anterior. Selecciona algunos datos de la columna*

description_y *como datos de "prueba" y verifica qué texto del conjunto de entrenamiento es el más similar a cada uno usando la distancia mínima en el espacio de componentes principales. ¿Coinciden con los del archivo original?*

De hecho, el nivel de accuracy que obtenía para una muestra fija y de tamaño 10 % con respecto a la muestra total, me permitió decidir el 'mejor' kernel string del inciso anterior. Fijando la semilla a 0, tomando un conjunto 'test' de 24 observaciones, y entrenando el kernel-pca, bajo las mismas condiciones del inciso anterior, se obtiene un ejemplo como el mostrado en la figura 5.2, por otro lado al considerar como conjunto test a toda la columna de 'description_y' para toda la muestra (las 2142 observaciones) se tiene una matriz de confusión como la que se muestra en el cuadro 5.1 donde vemos que el desempeño del clasificar es pobre pues el accuracy es inferior a .5. Por otro lado al usar el mismo conjunto train como test se logra una clasificación perfecta para cada observación, entonces concluimos de este ejercicio que el bajo nivel de accuracy sobre el conjunto 'test' de todas las observaciones en la columna 'description_y' tomando como train el conjunto de la columna 'description_x' hace notorio lo complejo que puede llegar a ser trabajar con datos poco estructurados como el texto, y exhibe la necesidad de hacer más contrastante la diferencia entre las densidades de ambos conjuntos de training y test, pues este bajo contraste es lo que hace difícil clasificar correctamente inclusive a los que sí se marcan como coincidentes en el conjunto test (y que inclusive en la muestra de train son mayoría).

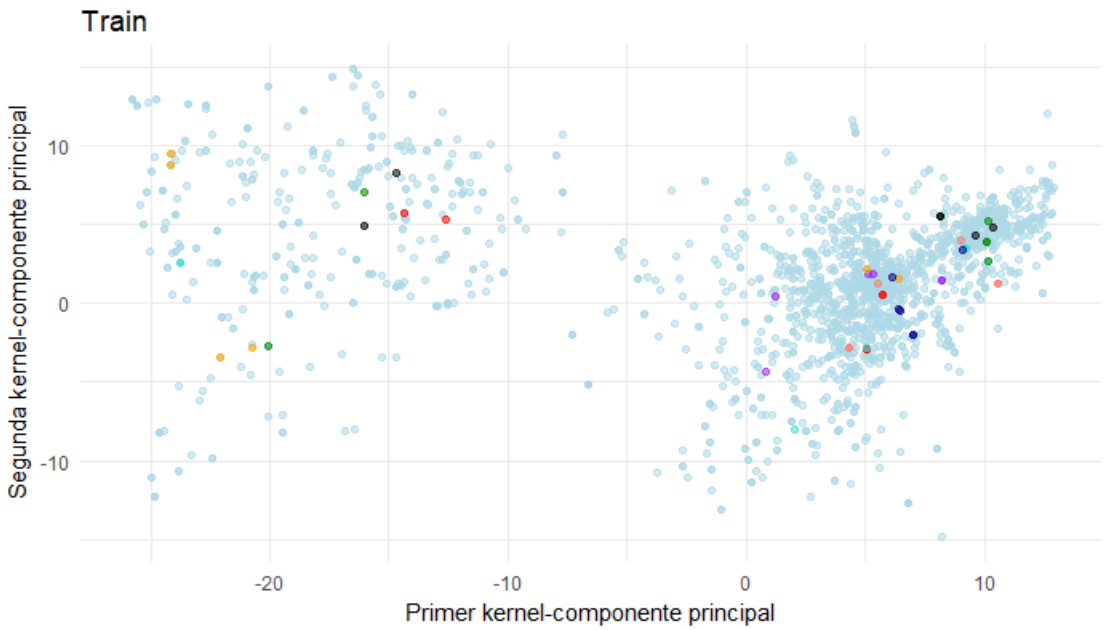

Figura 5.2: Ejemplo de clasificación de 24 observaciones tomando la distancia mínima euclidiana en el espacio de componentes principales de kernel-pca. Se presentan 8 colores, donde cada par de puntos (tanto la proyección del string test en el plano de las dos componentes principales, como el punto en el conjunto train más próximo a él) comparte color. Recordemos que este espacio es una proyección del espacio de todas las componentes principales, por lo que puntos del mismo color pueden parecer lejanos en esta proyección en particular.

	Predicción	
'same_security'	FALSE	TRUE
FALSE	.23	.02
TRUE	.52	23

Cuadro 5.1: Acurracy del método propuesto, utilizando como conjunto test a toda la columna 'description_y' y como train a toda la columna 'description_x'

En el anexo E se incluye el código respectivo de este análisis que acompaña a la tarea en el archivo ejercicio5.R

6. Anexo A

Código del análisis efectuado en el ejercicio 1.

```
setwd('C:\\Users\\fou-f\\Desktop\\MCE\\Second\\CienciaDeDatos\\tarea2')
genes <- read.csv('gene_expression_2classes.csv', stringsAsFactors = FALSE, header=FALSE)
pacientes <- t(as.matrix(genes))
pacientes <- as.data.frame(pacientes)
S <- cor(pacientes)
e.d <- eigen(S) #realizamos PCA sobre la matriz de correlacion
cumsum(e.d$values)[1:5]/sum(e.d$values)#checamos la varianza explicada por las primeras 5componentes
datos.rotados <- (as.matrix(scale(pacientes))) %*% e.d$vectors #rotamos los datos iniciales escalados
grupos <- kmeans(x = (pacientes), centers = 2, algorithm = 'Lloyd' )
plot(datos.rotados[,1], datos.rotados[,2], col= grupos$cluster, pch =20)
datos.rotados <- as.data.frame(datos.rotados)
clasificacion <- list('sano'=2, 'enfermo'=1)
clase <- unlist(names(clasificacion[grupos$cluster]))
library(ggplot2) #graficamos la proyeccion en las dos primeras componentes
ggplot(datos.rotados, aes(x=V1, y = V2,
                          color=clase)) +
  geom_point()+
  ggtitle('Kmeans sobre PCA (datos centrados)') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
cumsum(e.d$values[1:2])/sum(e.d$values)# pese a que la primer componente solo capta 8 % de la variacion
distancia <- scale((as.matrix(pacientes))) #buscamos cluster usando metodos jerarquicos
arbol <- hclust(dist(distancia, method = 'euclidean' ), method ="ward.D")
plot(arbol)
labels <- cutree(arbol, k=2)
library(dendextend) #pintamos el arbol con los clusters
arbol.feliz <- arbol %>% as.dendrogram %>%
  set("branches_k_color", k=2, c('purple','green4')) %>% set("branches_lwd", 1.2) %>%
  set("labels_colors", c(rep('purple',20), rep('green4',20))) %>%
  set("labels_cex", c(.9))
plot(arbol.feliz)
################inciso 2
R <- cor((as.matrix(pacientes)))
a <- heatmap(t(as.matrix(genes))) #el heatmap parece reconocer grupos de genes parecidos
image(t(as.matrix(genes))[a$rowInd, a$colInd]) #extraemos del heatmap los que parecen forman el grupo
      significativo
columnas <- tail(a$colInd, 111)
image(R)
#realizamos cluster jerarquico con matriz de similaridades inducida por la matriz
# de correlacion
R_dist <- diag(1000)-R
distancias <- as.dist(R_dist)
h <- hclust(distancias, method = 'ward.D')
plot(h)
labels<- cutree(h, h = 10)
freq <- as.data.frame(table(labels))
#decoramos el arbol
otro.arbol.feliz <- h %>% as.dendrogram %>%
  set("branches_k_color", k=2, c('purple','green4')) %>% set("branches_lwd", 1.2) %>%
  set("labels_colors", c(rep('purple',freq$Freq[2]),
                         rep('green4',freq$Freq[1]))) %>%
  set("labels_cex", c(.1))
plot(otro.arbol.feliz)
genes <- labels[labels==2] #extraemos lel segundo grupo del heatmap
columnas
genes.columnas <- gsub("V", "", names(genes))#extraemos solo el numero de columna
genes.columnas <- as.numeric(genes.columnas)
comparacion <- as.matrix(table(sort(genes.columnas), sort(columnas)))#lo comparo contra el grupo de menor
```

```
      cardinalidad encontrado
image(diag(111)-comparacion) ##color uniforme, son identicos los conjuntos
table(labels)
genes.columnas
######### Analisis analogo sin los genes importantes
test <- pacientes[,-(genes.columnas)]
S <- cor(test)
e.d <- eigen(S)
cumsum(e.d$values)[1:5]/sum(e.d$values)
datos.rotados <- (as.matrix(scale(test))) %*% e.d$vectors
grupos <- kmeans(x = scale(test), centers = 2, algorithm = 'Lloyd' )
plot(datos.rotados[,1], datos.rotados[,2], col= grupos$cluster, pch =20)
plot(datos.rotados[,1], datos.rotados[,2],
    col= c(rep('red',20), rep('blue',20)), pch =20)
datos.rotados <- as.data.frame(datos.rotados)
clasificacion <- list('sano'=2, 'enfermo'=1)
clase <- unlist(names(clasificacion[c(rep(2,20), rep(1,20))]))
library(ggplot2)
ggplot(datos.rotados, aes(x=V1, y = V2,
                          color=clase)) +
  geom_point()+
  ggtitle('Kmeans sobre PCA (datos centrados) sin genes importantes') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
cumsum(e.d$values[1:2])/sum(e.d$values)# pese a que la primer componente solo capta 8 % de la variacion
```

7. Anexo B

Implementación del algoritmo Kernel k-means en el lenguaje R y código necesario para generar las ilustraciones del ejercicio 2.

```
######### Implementacion de kernel k-means con shift basado en el paper:
#########Inderjit Dhillon, Yuqiang Guan and Brian Kulis.
#####A Unified view of Kernel k-means, Spectral Clustering and Graph Cuts.
Kernel.normal <- function(x,y,
                          #alpha = (.05/2)**.5) #kernel gaussiano
                          alpha = 0.0013984457316839 ) #para el caso de vinos
{
  return(exp(-sum((x-y)**2)/(2*alpha**2)))
}
Kernel.poli <- function(x,y,c = 1, d =2) #kernel polinomial
{
  return((sum(x*y)+c)**d)
}
Kernel.kmeans.init <- function(data, cols,  f=Kernel.normal )
{
  # data (data.frame): data set con las observaciones
  # cols (vector numeric): indices de las columnas para el calculo
  #de las distancias de 'data'
  #f (function): kernel a usar
  #se genera un dataframe con los pares de indices de las observaciones
  #con la finalidad de agilizar el calculo de la matriz de distancias 'MK'
```

```
# ESTA FUNCION ES UN CLOSURE, REGRESA OTRA FUNCION, funciona como un constructor de clase del paradigma POO
# LA FINALIDAD ES CALCULUAR LA MATRIZ DE KERNEL UNA SOLA VEZ Y PROBAR DIFERENTES VALORES DE PARAMETROS
indices <- 1:dim(data)[1]
index <- data.frame(x1 = rep(indices,dim(data)[1]),
                    x2=rep(indices,each=dim(data)[1]))
library(parallel)#para incrementar la velocidad usaremos calculo multicore
#comienza calculo de la matriz kernel entre todos los pares de observaciones
kernel <- mclapply(FUN=function(i, cols){
  f(x=data[index[i,1],][cols],
    y=data[index[i,2],][cols])
}, X = 1:dim(index)[1], cols,
mc.cores = detectCores()-2 )
Kernel <- unlist(kernel)
MK <- as.matrix(Kernel)
dim(MK) <-  rep(dim(data)[1], 2)
#termina calculo de matriz de kernel
function(data, sigma,t, k )
{
  #data (data.frame) con las observaciones a clasificar
  #sigma (numeric): shift mencionado en el paper
  #t (numeric): numero de iteraciones
  #k (numeric): numero de clusters
  data$cluster <- sample(1:k,dim(data)[1], replace = TRUE)#asignacion inicial aleatoria
  for(x in 1:t)
  {
    #siguiendo el paper citado realizamos las iteraciones
    #hacemos uso del shift
    nuevo.cluster <- mclapply(FUN=function(i)
    {
      #parte del calculo que no depende del shift
      d <- rep(-1, k)
      for(z in 1:k)
      {
        j <- which(data$cluster == z)
        d[z]<- MK[i,i] - 2*sum(MK[i, j])/length(j) +
          sum(MK[j,j])/(length(j))**2
      }
      pis<- data.frame(table(data$cluster))
      # suma del shift a las entradas correspondientes
      for(z in 1:k)
      {
        if(z !=data[i,'cluster'])
        {
          d[z] <- d[z] + sigma +sigma/pis[z,'Freq']
        }else{
          d[z] <- d[z] + sigma -sigma/pis[data[i,'cluster'],'Freq']
        }
      }
      (1:k)[which.min(d)]#nueva asignacion de cluster para la observacion
    }, X =1:dim(data)[1], mc.cores = detectCores()-2
    )
    data$cluster <- unlist(nuevo.cluster) #juntamos los resultados
  }
  return(data$cluster) #cluster finales
 }
}
#################################################################
```

```r
set.seed(0)
#####simulacion de datos parecidos a los del paper mencionado
#####son dos circunferencias con centro (.5,.5) y radios 1 y 4
#####se agrega en cada eje ruido ~ N(0,sigma=1/10 ) y N(0,1/10)
r <- 1 #radio
n <- 100 #la cuarta parte del numero de puntos que se van a generar
#se genera la primer circunferencia con ruido
x <- seq(-r, r, length=n)
y1 <- sqrt(r**2-x**2) + rnorm(n,0,r/10)
y2 <- -sqrt(r**2-x**2) - rnorm(n,0,r/10)
m.a1 <- data.frame(x=rep(x+.5, 2), y = c(y1+.5,y2+.5), clase=1)
#se genera la segunda circunferencia con ruido
r <- 4
n <- 100
x <- seq(-r, r, length=n)
y1 <- sqrt(r**2-x**2) + rnorm(n,0,r/40)
y2 <- -sqrt(r**2-x**2) - rnorm(n,0,r/40)
m.a2 <- data.frame(x=rep(x+.5, 2), y = c(y1+.5,y2+.5), clase=2)
m.a <- rbind(m.a1, m.a2) #nuestro primer conjunto de prueba
library(ggplot2)
ggplot(m.a,#visualizamos nuestro primer conjunto de prueba
       aes(x=x, y=y, color = factor(clase))) + geom_point() +
    theme_minimal() + theme(legend.position='none') +
    ggtitle('Muestra aleatoria generada (400 obs)') +xlab('') + ylab('')
label<- kmeans(m.a, centers = 2) #comparamos el desempeno de kmeans en vista
#                              #de que apriori sabemos que son 2 grupos
p1 <- ggplot(m.a,#visualizamos nuestro primer conjunto de prueba
       aes(x=x, y=y, color = factor(label$cluster))) + geom_point() +
    theme_minimal() + theme(legend.position='none') +
    ggtitle('Agrupamiento de kmeans (accuracy .49 %)') +xlab('') + ylab('')
#visualizamos los resultados
             #de clasificacion usando kmeans
sum(diag(as.matrix(table(m.a$clase, label$cluster))))/dim(m.a)[1] #se calcula el accuracy que es de .265,
      .735,.4975
#uso sobre
Kernel.kmeans.simu <- Kernel.kmeans.init(m.a, cols=1:2)
labels <- Kernel.kmeans.simu(data= m.a, sigma = -1, t = 20, k =2)
m.a$cluster <- labels
p3 <- ggplot(m.a,#visualizamos nuestro primer conjunto de prueba
            aes(x=x, y=y, color = factor(cluster))) + geom_point() +
    theme_minimal() + theme(legend.position='none') +
    ggtitle('Agrupamiento de kernel.kmeans (accuracy .72 %)') +xlab('') + ylab('')
sum(diag(as.matrix(table(m.a$clase,m.a$cluster))))/dim(m.a)[1]
# accuracy .3275 con shif de -1, 2175 con -.05, .47 con -3, .4875 con -5 totos los casos con t = 10 , .68 con 0
#t=20: .72 con -1,.7175 con  -.05, -.5575 con -3, -.4525 con -5, .2575 con 0
#t=100, .69  con-1,.2525 con -.05,.5125 con -3,.4925 con -5, .245 con 0
####################
#########Comparacion de resultados usando la implementacion de kernlab
library(kernlab)
kkmeans.m.a <- kkmeans(as.matrix(m.a[,1:2]), centers=2 ,
                       kernel = "rbfdot",
                       kpar = list(sigma = (.05/2)**.5),
                       alg="kkmeans")#mismo kernel y parametro del kernel
sum(diag(as.matrix(table(m.a$clase, kkmeans.m.a@.Data))))/dim(m.a)[1]#accuracy de 1
p2 <- ggplot(m.a, aes(x=x, y=y, color = factor(kkmeans.m.a@.Data)))+
    geom_point() +theme_minimal()+theme(legend.position='none') +
    ggtitle('Agrupamiento perfecto de kkmeans')+xlab('')+ylab('')
library(ggpubr)
ggarrange(p1,p2,p3, nrow = 3)
```

```
##########
setwd('/home/fou/Desktop')
vinos <- read.csv('wine_quality.csv')
set.seed(0)
muestra.vinos <- sample(1:dim(vinos)[1], 650)
vinos <- vinos[muestra.vinos,]
vinos$cluster <- kmeans(vinos[,2:12], centers=10)$cluster
confusion <- as.matrix(table(vinos$quality, vinos$cluster))
sum(diag(confusion))/dim(vinos)[1]#.09
pca.vinos <- princomp(vinos[,2:12], cor = TRUE)
pca.vinos.prin <- pca.vinos$scores[,1:2]
pca.vinos.prin <- as.data.frame(pca.vinos.prin)
p1 <- ggplot(pca.vinos.prin,#visualizamos nuestro primer conjunto de prueba
             aes(x=Comp.1, y=Comp.2, color = factor(vinos$cluster))) + geom_point() +
    theme_minimal() + theme(legend.position='none') +
    ggtitle('Agrupamiento de kmeans (accuracy .12 %)') +
    xlab('Primer componente principal') + ylab('Segunda componente principal')
#
t1 <- Sys.time()
Kernel.kmeans.vinos <- Kernel.kmeans.init(vinos, cols=2:12)
t1 <- Sys.time()-t1
labels <- Kernel.kmeans.vinos(data=vinos, sigma = -1, t=20, k=10)
vinos$cluster <- labels
sum(diag(as.matrix(table(vinos$quality,vinos$cluster))))/dim(vinos)[1] #acurracy con shiff -1 .16
p3 <- ggplot(pca.vinos.prin,#visualizamos nuestro primer conjunto de prueba
       aes(x=Comp.1, y=Comp.2, color = factor(vinos$cluster))) + geom_point() +
    theme_minimal() + theme(legend.position='none') +
    ggtitle('Agrupamiento de Kernel.kmeans (accuracy .16 %)') +
    xlab('Primer componente principal') + ylab('Segunda componente principal')
##########################################################
##############comparacion para el dataset de vinos
kkmeans.vinos <- kkmeans(x=as.matrix(vinos[,2:12]), centers=10 ,
                         kernel = "rbfdot",
                         alg="kkmeans")
sum(diag(as.matrix(table(vinos$quality, kkmeans.vinos@.Data))))/dim(vinos)[1]#.12 con el sigma que yo doy
      .1078, .1237 con el default que vale 0.00153861608749537
#visualizamos el agrupamiento en dos dimensiones
p2 <- ggplot(pca.vinos.prin, aes(x=Comp.1, y=Comp.2,
                          color = factor(kkmeans.vinos@.Data)))+
    geom_point()+theme_minimal()+ theme(legend.position='none') +
    ggtitle('Agrupamiento de kkmeans (accuracy .12 %)') +
    xlab('Primer componente principal') + ylab('Segunda componente principal')
ggarrange(p1,p2,p3, ncol= 3)
```

8. Anexo C

Código con el cual se realizó el análisis del ejercicio 3, al igual que las figuras.

```
setwd('C:\\Users\\fou-f\\Desktop\\MCE\\Second\\CienciaDeDatos\\tarea2\\data_fruits_tarea')
imagenes <- dir()
library(imager)
L2 <- function(x,y) #funcion para medir distancia entre pixeles
{
```

```
  # x (vector-numeric): pixel de la imagen
  # y (vector-numeric): pixel con valores (1,1,1)
  return(sqrt(sum((x-y)**2)))
}
index <- length(imagenes)
medianas <- matrix(rep(-1,3*index), nrow = index) #se reserva espacio para guardar las medianas de los incisos
      1 a 3
##########inicia etapa de preprosemiento la cual consiste en trabajar
##########solo con pixeles que difieren en una pequena distancia del pixel blanco
for(i in 1:index)
{
  fruta <- load.image(file=imagenes[i])
  print(imagenes[i])
  copia <- fruta
  # for(j in 1:100)#para cada pixel se mide su distancia con el pixel blanco
  # {
  #   for(k in 1:100)
  #   {
  #     if(  L2(copia[j,k,,1:3], rep(1.,3)) <=0.02037707)
  #     {
  #       #aproximadamente si el pixel dista en menos de 3 valores del blanco se elimina de la imagen
  #       copia[j,k,1:3] <- rep(NA, 3)
  #     }
  #   }
  # }
  #plot(copia)
  for(j in 1:3)
  {
    #se calculan las medianas requeridas solamente sobre los pixeles
    #considerados como informativos de la propia imagen
    mediana <- median(copia[,,j], na.rm = TRUE)
    medianas[i, j] <- mediana
  }
  remove(copia)
  remove(fruta)
}
#####se termina preprocesamiento
colnames(medianas) <- c('r', 'g', 'b')
#revise si habia diferencias en el encoding, porque la funcion da puntos en [0,1]
#mientras yo esperaba enteros en [0,255] y no hubo diferencias
reduce.a.medianas.discreta <- function(index)
{
  fruta <- load.image(file=imagenes[index])
  fruta[,,1:3] <- fruta[,,1:3]*255
  medianas <- apply(fruta[,,1:3], 3, median)
  medianas <- matrix(medianas, ncol = 1)
  row.names(medianas) <- c('r', 'g', 'b')
  return(medianas)
}
#se manipulan las medianas para convertirlas en un data.frame facil de manjer en las
#visualizaciones siguientes
imagenes.medianas <- medianas
df.imagenes.medianas <- as.data.frame(imagenes.medianas)
#no.normalizados <- lapply(1:length(imagenes), FUN= reduce.a.medianas.discreta)
#no.normalizados <- do.call("cbind", no.normalizados)
#df.no.normalizados <- as.data.frame(no.normalizados)
#df.no.normalizados <- as.data.frame(t(as.matrix(df.no.normalizados)))
#se obtienen los nombres de las imagenes provenientes del directorio
library(stringr)
```

```
imagenes.nombres <−str_split(imagenes,regex("[:digit:]"),simplify = T)[,1]# "Eliminar de un digito hacia adelante
    "
imagenes.nombres <− str_split(imagenes.nombres, regex("$"),simplify = T)[,1]  # Elimnar los "" finales
imagenes.nombres <−str_split(imagenes.nombres, regex("r$"),simplify = T)[,1]  # "Eliminar de un "_r" hacia
    adelante"
df.imagenes.medianas$tipo <− imagenes.nombres #se agregan al conjunto de datos los nombres 'tipos' de frutas
df.imagenes.medianas$tipo <− factor(df.imagenes.medianas$tipo)
df.imagenes.medianas$tipo2 <− df.imagenes.medianas$tipo
#en la siguiente seccion se crea una columna que agrupa por tipo de fruta sin importar su orientacion
levels(df.imagenes.medianas$tipo2)  <− c(rep('Apple_Braeburn',2),
        'Apple_Golden', rep('Apple_Granny_Smith', 2),
        rep('Apricot', 2), rep('Avocado', 2), rep('Carambula', 2),
        rep('Cherry', 2), rep('Huckleberry', 2), rep('Kiwi', 2), rep('Orange', 2),
        rep('Peach', 2), rep('Pineapple', 2), rep('Strawberry', 2))

#se definen colores para las visualizaciones de los conjuntos
colores <− c('red', 'green', 'green4', 'pink4',
          'black', 'yellow', 'purple', 'navy',
          'brown', 'orange', 'salmon',
          'darkgoldenrod1', 'magenta')
#df.no.normalizados$tipo <− df.imagenes.medianas$tipo
#df.no.normalizados$tipo2 <− df.imagenes.medianas$tipo2
library(plotly) #visualizacion en 3D de las imagenes representadas por las medianas en cada canal
p1 <− plot_ly(df.imagenes.medianas, x = ~r, y = ~g,
          z = ~b, color = ~tipo2,
          colors = colores) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'Red'),
                      yaxis = list(title = 'Green'),
                      zaxis = list(title = 'Blue')))
p1
#se realiza PCA sobre las medianas PCA con matriz de covarianzas
PCA <− princomp(df.imagenes.medianas[,1:3], cor = FALSE, scores = TRUE)
valores.pro <− PCA$sdev**2
cumsum(valores.pro)/sum(valores.pro) #las dos primeras componentes representan 96.73 % de la varianza total
datos.rotados <− as.matrix(df.imagenes.medianas[,1:3]) %* %PCA$loadings#se rotan los datos
datos.rotados <− as.data.frame(datos.rotados)
datos.rotados$tipo <− df.imagenes.medianas$tipo
datos.rotados$tipo2 <− df.imagenes.medianas$tipo2
datos.rotados$tipo3 <− colores[datos.rotados$tipo2]
p1 <− ggplot(datos.rotados, aes(x=Comp.1, y = Comp.2,
                        color=tipo2)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('PCA imagenes') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
# se realiza kernel−pca
library(kernlab)
set.seed(0)
kernel.pca <− kpca(~., data=df.imagenes.medianas[,1:3], kernel="rbfdot", kpar=list(sigma=1/90))
datos.kernel <− as.data.frame(rotated(kernel.pca))
eigen.values <− eig(kernel.pca)
cumsum(eigen.values)/sum(eigen.values)#las dos primeras componentes explican el 96 % de la varianz
datos.kernel$tipo2 <− df.imagenes.medianas$tipo2
p2 <− ggplot(datos.kernel, aes(x=V1, y = V2,
                        color=tipo2)) +
  geom_point()+
  scale_color_manual(values=colores)+
```

```
  ggtitle('Kernel PCA (imagenes)') + theme_minimal() +
  xlab('Kernel−Primer componente principal') +
  ylab('Kernel−Segunda componente principal')
p2
library(ggpubr)
ggarrange(p1, p2, ncol = 2, nrow = 1)
#se realiza kmeans y kernel−kmeans
set.seed(0)
df.imagenes.medianas$kmeans <− factor(kmeans((df.imagenes.medianas[,1:3]), centers=length(colores), nstart =
      100 )$cluster)
table(df.imagenes.medianas$tipo2, df.imagenes.medianas$kmeans)#error de 0.2461538
set.seed(0)
kk <− kkmeans(as.matrix((df.imagenes.medianas[,1:3])), centers = 13,
       kernel='rbfdot',  kpar = list(sigma = 90))
df.imagenes.medianas$kkmeans <− factor(kk@.Data)
k1 <− ggplot(datos.rotados, aes(x=Comp.1, y = Comp.2,
                      color=df.imagenes.medianas$kmeans)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('Grupos usando kmeans (imagenes)') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
kk1 <− ggplot(datos.rotados, aes(x=Comp.1, y = Comp.2,
                       color=df.imagenes.medianas$kkmeans)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('Grupos usando kernel−kmeans (imagenes)') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
table( df.imagenes.medianas$tipo2, df.imagenes.medianas$kkmeans)
402/1300 #sigma de 100
473/1300 #sigma de 60
452/1300 #sigma de 200
504/1300 #sigma de 80
388/1300 #sigma de 90
475/1300 #sigma de 95
482/1300 #sigma de 85
408/1300 #sigma de 50
ggarrange(p1, k1, kk1, ncol = 2, nrow = 2)

############ inciso d imagenes.nombres
##de nuevo el preprocesamiento de quitar pixeles cercanos al blanco
medianas.HSV <− matrix(rep(−1,9∗index), nrow = index) #se reserva espacio para guardar las medianas de los
      incisos 1 a 3
for(i in 1:index)
{
  fruta <− load.image(file=imagenes[i])
  print(imagenes[i])
  copia <− fruta
  for(j in 1:100)#para cada pixel se mide su distancia con el pixel blanco
  {
    for(k in 1:100)
    {
      if(  L2(copia[j,k,1,1:3], rep(1.,3)) <=0.02037707)
      {
        #aproximadamente si el pixel dista en menos de 3 valores del blanco se elimina de la imagen
        copia[j,k,1, 1:3] <− rep(NA, 3)
      }
    }
```

```
  }
  #plot(copia)
  copia <- RGBtoHSV(copia)
  for(j in c(1,4,7))
  {
    #se calculan las medianas requeridas solamente sobre los pixeles
    #considerados como informativos de la propia imagen
    canal <- round(j/3,0)+1
    mediana <- quantile(copia[,,1,canal], na.rm = TRUE, probs =c(.25,.5,.75)  )
    medianas.HSV[i, j:(j+2) ] <- mediana
  }
  remove(copia)
  remove(fruta)
}
###### se termina preprocesamiento
#la siguinete funcion la utilice en su momento para extraer los
# cuartiles sin considerar el preprocesamiento dd quitar los pixeles cercanos al
# pixel blanco, como se obtuvieron resultados pobres se abandono este esquema
# reduce.a.medianas.HSV <- function(index)
# {
#   fruta <- load.image(file=imagenes[index])
#   fruta <- RGBtoHSV(fruta)
#   mediana <- apply(fruta[,,1,1:3], 3, function(x){
#     quantile(x, probs = c(.25, .5, .75 ))
#   })
#   medianas <- matrix(mediana, byrow = TRUE, ncol = 9)
#   colnames(medianas) <- c('H.25','H.5', 'H.75',
#                           'S.25','S.5', 'S.75',
#                           'V.25','V.5', 'V.75')
#   row.names(medianas) <- index
#   return(medianas)
# }
imagenes.medianas.hsv <- medianas.HSV
#se manipulan los cuartiles para convertirlas en un data.frame facil de manjer en las
#visualizaciones siguientes
  #se copian columnas  on el tipo de fruta sin importar su orientacion y consideracndo su orientacion
df.imagenes.medianas.hsv <- as.data.frame(imagenes.medianas.hsv)
df.imagenes.medianas.hsv$tipo1 <- df.imagenes.medianas$tipo
df.imagenes.medianas.hsv$tipo2 <- df.imagenes.medianas$tipo2
#se realiza PCA sobre las medianas PCA con matriz de covarianzas
PCA <- princomp(df.imagenes.medianas.hsv[,1:9], cor = FALSE, scores = TRUE)
valores.pro <- PCA$sdev**2
cumsum(valores.pro)/sum(valores.pro) #las dos primeras componentes representan 99.9 % de la varianza total
datos.rotados <- as.matrix(df.imagenes.medianas.hsv[,1:9]) %* %PCA$loadings#se rotan los datos
datos.rotados <- as.data.frame(datos.rotados)
datos.rotados$tipo <- df.imagenes.medianas.hsv$tipo
datos.rotados$tipo2 <- df.imagenes.medianas.hsv$tipo2
p1 <- ggplot(datos.rotados, aes(x=Comp.1, y = Comp.2,
                                color=tipo2)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('PCA imagenes') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
p1
# se realiza kernel-pca
library(kernlab)
set.seed(0)
kernel.pca <- kpca(~., data=df.imagenes.medianas.hsv[,1:9],
```

```
                  kernel="rbfdot", kpar=list(sigma=.001*4))#.001*5
datos.kernel <- as.data.frame(rotated(kernel.pca))
eigen.values <- eig(kernel.pca)
cumsum(eigen.values)/sum(eigen.values)#las dos primeras componentes explican el 16.37 % de la varianz
0.3597184
datos.kernel$tipo2 <- df.imagenes.medianas$tipo2
p2 <- ggplot(datos.kernel, aes(x=V1, y = V2,
                               color=tipo2)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('Kernel PCA (imagenes)') + theme_minimal() +
  xlab('Kernel-Primer componente principal') +
  ylab('Kernel-Segunda componente principal')
p2
ggarrange(p1, p2, ncol = 2, nrow = 1)
#se realiza kmeans y kernel-kmeans
set.seed(0)
df.imagenes.medianas.hsv$kmeans <- factor(kmeans((df.imagenes.medianas.hsv[,1:9]),
                                 centers=length(colores), nstart = 100 )$cluster)
table(df.imagenes.medianas.hsv$tipo2, df.imagenes.medianas.hsv$kmeans)#error de 0.2461538
401/1300 #error kmeans usando cuartiles
set.seed(0)
kk <- kkmeans(as.matrix(df.imagenes.medianas.hsv[,1:9]),
              centers = 13,
              kernel='rbfdot',  kpar = list(sigma = .04))
df.imagenes.medianas.hsv$kkmeans <- factor(kk@.Data)
k1 <- ggplot(datos.rotados, aes(x=Comp.1, y = Comp.2,
                               color=df.imagenes.medianas.hsv$kmeans)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('Grupos usando kmeans (imagenes)') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
kk1 <- ggplot(datos.rotados, aes(x=Comp.1, y = Comp.2,
                                 color=df.imagenes.medianas.hsv$kkmeans)) +
  geom_point()+
  scale_color_manual(values=colores)+
  ggtitle('Grupos usando kernel-kmeans (imagenes)') + theme_minimal() +
  xlab('Primer componente principal') +
  ylab('Segunda componente principal')
table( df.imagenes.medianas.hsv$tipo2, df.imagenes.medianas.hsv$kkmeans)
#con los cuartiles
#clasifica muy mal  #sigma de 100, 300,10
732/1300#sigma de 1
516/1300 #sigma de .1
475/1300 #sigma de .01
422/1300 #sigma de .05
434/1300 #sigma de .02
460/1300 #sigma de .03
422/1300 #sigma de .04
467/1300 #sigma de .06
540/1300 #sigma de .09
ggarrange(p1, k1, kk1, ncol = 2, nrow = 2)
#############contraste
colnames(df.imagenes.medianas.hsv) <- c('H.25', 'H.5', 'H.75',
                                        'S.25', 'S.5', 'S.75',
                                        'V.25', 'V.5', 'V.75',
                                        'tipo1', 'tipo2', 'kmeans', 'kkmeans')
library(plotly) #visualizacion en 3D de las imagenes representadas por las medianas en cada canal
```

```
p1 <- plot_ly(df.imagenes.medianas.hsv, x = ~H.75, y = ~S.75,
              z = ~V.75, color = ~tipo2,
              colors = colores) %>%
  add_markers() %>%
  layout(scene = list(xaxis = list(title = 'H.75'),
                      yaxis = list(title = 'S.75'),
                      zaxis = list(title = 'V.75')))
p1
```

9. Anexo D

Código respectivo para el análisis del ejercicio 4.

```
directorio <-'C:\\Users\\fou-f\\Desktop\\MCE\\Second\\CienciaDeDatos\\tarea2\\movie_reviews\\txtfiles'
setwd(directorio)
dir()
#primero veamos las palabras de las opiniones negativas
#para contrastarlas con las palabras de las opiniones positivas y lograr
#realzar el contraste entre ambos grupos quitando las palabras que aparecen en
#ambas bolsas de palabras
preprocesamiento <- function(x)
{
  #esta funcion tiene como finalidad extraer todas las palabras
  #regresa un dataset con las frecuencias acumuladas por palabras en el corpus
  # y una matriz de treminos-documentos
  # x (path): path en donde se encuentran lojados en disco duro los documentos
  library(tm)
  negativos <- Corpus(DirSource(x,recursive=TRUE),
              readerControl=list(language="en_US"))
  #preprosamiento
  negativos <- tm_map(negativos,stripWhitespace)
  negativos <- tm_map(negativos,removeNumbers) #tal vez los numero sean inportantes
  negativos <- tm_map(negativos,content_transformer(tolower))
  negativos <- tm_map(negativos,removePunctuation) #talvez los emoticones tambien sean importantes
  negativos <- tm_map(negativos,removeWords,stopwords("english"))
  negativos <- tm_map(negativos,stemDocument)
  ## obtiene matriz de terminos
  matriz.neg <- TermDocumentMatrix(negativos)#,control=list(minDocFreq=100))
  m.neg <- as.matrix(matriz.neg)
  v <- sort(rowSums(m.neg),decreasing=TRUE)
  d.neg <- data.frame(word = names(v),freq=v)
  d.neg <- d.neg[order(d.neg$word),]
  return(list(frecuencias= d.neg, mtd=m.neg ))
}
#########################
negativos <- preprocesamiento(dir()[1]) #preprocesamos los documentos marcados como negativos
positivos <- preprocesamiento(dir()[2]) #preproseamiento de los documentos marcados como positivos
# identificamos las palabras comunes a ambos conjuntos
comunes <- merge(negativos[['frecuencias']],positivos[['frecuencias']]  , by ='word')
#checamos la correlacion de las frecuencias de las palabras en ambos conjuntos
#primero visualmente
library(ggplot2)
ggplot(comunes, aes(x = freq.x, y = freq.y, alpha=.0001 ))+
```

```
  geom_point()+theme_minimal()+ theme(legend.position="none")+
  xlab('Conjunto de palabras en la bolsa "negativas"')+
  ylab('Conjunto de palabras en la bolsa "positivas"')+
 ggtitle('Frecuencia de palabas comunes en ambos conjuntos de resenas ')
#parece indicar correlacion positiva
#hacemos un test de significancia
cor.test(comunes$freq.x, comunes$freq.y, method = 'pearson', alternative='two.sided')
#como el p-value es casi cero rechazamos la hipotesis nula de no correlacion
#asumimos quue la frecuencia de las palabras es aproximadamente la misma
#por lo que las descartamos del analisis pues solo incrementarian la dimencion de la tarea y no
#ayudan a diferenciar el sentimiento
##################################################
todos <- preprocesamiento(dir()) #todos los datos
utiles <- todos[['mtd']]
#primer filtro quitamos las palabras que no aportan informacion
utiles <- utiles[!(row.names(utiles) %in% as.character(comunes$word)),  ]
df.m <- as.data.frame(t(utiles))
d <- todos[['frecuencias']]
d <- d[!(d$word %in% as.character(comunes$word)),]
#procedemos a eliminar las palabras con menor frecuencia
poquitas <- d$word[d$freq>5] #determinamos las palabras comunes y con una frecuencia mayor a 6
#library(wordcloud)
wordcloud(comunes$word, comunes$freq.x)
utiles <- utiles[(row.names(utiles) %in% as.character(poquitas)),  ]
library(dplyr) #fiiltramode ugual manera la tabla de frecuuencias
df.m <- as.data.frame(t(utiles))
v <- sort(rowSums((utiles)),decreasing=TRUE)
d <- data.frame(word = names(v),freq=v)
#nos quedamos con 443 palabras unicamente
##########seccion un usar PCA para distinguir las recomendaciones postivas de las negativas
#PCA sobre matriz de terminos
R <- cor(df.m)
det(R)
pca <- eigen(R)
cumsum(pca$values)[1:20]/sum(pca$values) #las dos primeras componentes solo explican el 2.8 % de la varianza
    total
rotacion <- pca$vectors
datos.rotados <- as.matrix(df.m) %*% rotacion #rotamos los datos
#graficamos las resenas en el espacio de componentes principales
datos.rotados <- as.data.frame(datos.rotados)
datos.rotados$sentimiento <- 'negativo'
datos.rotados$sentimiento[501:1000] <- 'Positivo'
p1 <- ggplot(subset(datos.rotados, sentimiento=='negativo'),
      aes(x = V1, y = V2, color=sentimiento, alpha=.01 ))+
  geom_point()+theme_minimal()+
  xlab('Primer componente principal')+
  ylab('Segundo componente principal')+
  ggtitle('Resenas negativas proyectadas en las dos primeras componentes principales')
p2 <- ggplot(subset(datos.rotados, sentimiento=='Positivo'),
      aes(x = V1, y = V2, color=sentimiento, alpha=.01 ))+
  geom_point()+theme_minimal()+
  xlab('Primer componente principal')+
  ylab('Segundo componente principal')+
  ggtitle('Resenas positivas proyectadas en las dos primeras componentes principales')
p3 <- ggplot(datos.rotados, aes(x = V1, y = V2, color=sentimiento, alpha=.0001 ))+
  geom_point()+ theme_minimal()+
  xlab('Primer componente principal')+
  ylab('Segundo componente principal')+
  ggtitle('Todas las resenas proyectadas en las dos primeras componentes principales') +
```

```
  ylim(c(-1/10,1/10)) +xlim(c(-1/10,1/10))
library(ggpubr)
ggarrange(p1, p2, p3, ncol = 1, nrow = 3)
datos.rotados <- datos.rotados %>%
  mutate(etiquetaNegativo = !( (V1>=0) & (V2<=0)))
table(datos.rotados$sentimiento, datos.rotados$etiquetaNegativo)
############segundo inciso
# procedemos de manera analoga utilizando los datos que ya calculamos,
#primero determinamos las palabras en ambos conjuntos pero agrupados de 5 en 5
#primero agrupamos las palabras en comun resultaron ser las mismas
# matriz.neg <- negativos[['mtd']]
# matriz.neg <- as.data.frame(t(matriz.neg))
# k <- 5
# matriz.neg$grupo <- factor(rep(1:(dim(matriz.neg)[1]/k), each=k))
# library(dplyr)
# test <- matriz.neg %>% group_by(grupo) %>% summarise_all(funs(sum))
# test <- as.matrix(as.numeric(test[,2:15622]))
# v <- apply(test[,-1], 2, sum)
# d.neg <- data.frame(word = names(v),freq=v)
# matriz.pos <- positivos[['mtd']]
# matriz.pos <- as.data.frame(t(matriz.pos))
# matriz.pos$grupo <- factor(rep(1:(dim(matriz.pos)[1]/k), each=k))
# test2 <- matriz.pos %>%group_by(grupo) %>%summarise_all(funs(sum))
# v <- apply(test2[,-1], 2, sum)
# d.pos <- data.frame(word = names(v),freq=v)
# #por fin obtenemos las palabra comunes
# comunes <- merge(d.pos, d.neg  , by ='word') #las mismas 9477 palabras
##################################################
todos <- preprocesamiento(dir()) #todos los datos
utiles <- todos[['mtd']]
#primer filtro quitamos las palabras que no aportan informacion
utiles <- t(utiles)
k <- 5 #numero de agrupacion
utiles <- as.data.frame(utiles)
utiles$grupo <- factor(rep(1:(dim(utiles)[1]/k), each=k))
test <- utiles %>% group_by(grupo) %>% summarise_all(funs(sum))
dim(test)
head(test[,1:5])
v <- apply(test[,-1], 2, sum)
d <- data.frame(word = names(v),freq=v)
utiles.fino <- test[, !(colnames(test) %in% as.character(comunes$word))  ]
#procedemos a eliminar las palabras con menor frecuencia
poquitas <- d$word[d$freq>5] #determinamos las palabras comunes y con una frecuencia mayor a 6
utiles.fino <- utiles.fino[, (colnames(utiles.fino) %in% as.character(poquitas))  ]
df.m <- as.data.frame(utiles.fino)
v <- apply(utiles.fino[,-1], 2, sum)
d <- data.frame(word = names(v),freq=v)
#nos quedamos con 442 palabras unicamente
##########seccion un usar PCA para distinguir las recomendaciones postivas de las negativas
#PCA sobre matriz de terminos
R <- cor(df.m)
det(R)
pca <- eigen(R)
cumsum(pca$values)[1:20]/sum(pca$values) #las dos primeras componentes solo explican el 4.7% de la varianza
      total
rotacion <- pca$vectors
datos.rotados <- as.matrix(df.m) %*%rotacion #rotamos los datos
#graficamos las resenas en el espacio de componentes principales
datos.rotados <- as.data.frame(datos.rotados)
```

```
datos.rotados$sentimiento <- 'negativo'
datos.rotados$sentimiento[101:200] <- 'Positivo'
p1 <- ggplot(subset(datos.rotados, sentimiento=='negativo'),
             aes(x = V1, y = V2, color=sentimiento, alpha=.01 ))+
  geom_point()+theme_minimal()+
  xlab('Primer componente principal')+
  ylab('Segundo componente principal')+
  ggtitle('Resenas negativas proyectadas en las dos primeras componentes principales')
p2 <- ggplot(subset(datos.rotados, sentimiento=='Positivo'),
             aes(x = V1, y = V2, color=sentimiento, alpha=.01 ))+
  geom_point()+theme_minimal()+
  xlab('Primer componente principal')+
  ylab('Segundo componente principal')+
  ggtitle('Resenas positivas proyectadas en las dos primeras componentes principales')
p3 <- ggplot(datos.rotados, aes(x = V1, y = V2, color=sentimiento, alpha=.0001 ))+
  geom_point()+ theme_minimal()+
  xlab('Primer componente principal')+
  ylab('Segundo componente principal')+
  ggtitle('Todas las resenas proyectadas en las dos primeras componentes principales')
ggarrange(p1, p2, p3, ncol = 1, nrow = 3)
datos.rotados <- datos.rotados %>%
  mutate(etiquetaNegativo = !( (V1<=0) & (V2<=0)))
table(datos.rotados$sentimiento, datos.rotados$etiquetaNegativo)
```

10. Anexo E

Código respectivo para el análisis del ejercicio 5.

```
setwd('C:\\Users\\fou-f\\Desktop\\MCE\\Second\\CienciaDeDatos\\tarea2')
strings <- read.csv('train_stock.csv', stringsAsFactors = FALSE)
## veamos la longitud en palabras de los clasificados como iguales
iguales <- subset(strings, same_security==TRUE) #sleccionamos solo los etiquetados como semejantes
x1.len <- lapply(FUN= function(i){
    n <- strsplit(as.character(iguales[i,'description_x']),"  " )
    n <- unlist(n)
    n.len <- length(n)
    n.len
}, X =(1:dim(iguales)[1])) # checamos la longitud en palabras de la descripcion x
x2.len <- lapply(FUN= function(i){
  n <- strsplit(as.character(iguales[i,'description_y']),"  " )
  n <- unlist(n)
  n.len <- length(n)
  n.len
}, X =(1:dim(iguales)[1])) # checamos la longitud en palabras de la descripcion y
x1.len <- unlist(x1.len)
x2.len <- unlist(x2.len)
(table(x1.len, x2.len))# 4 es el mayor
## veamos la longitud en palabras de los clasificados como diferentes
diferentes <- subset(strings, same_security==FALSE)
y1.len <- lapply(FUN= function(i){  # checamos la longitud en palabras de la descripcion x
  n <- strsplit(as.character(diferentes[i,'description_x']),"  " )
  n <- unlist(n)
  n.len <- length(n)
```

```
  n.len
}, X =1:dim(diferentes)[1])
y2.len <- lapply(FUN= function(i){ # checamos la longitud en palabras de la descripcio y
  n <- strsplit(as.character(diferentes[i,'description_y']),"  " )
  n <- unlist(n)
  n.len <- length(n)
  n.len
}, X =1:dim(diferentes)[1])
y1.len <- unlist(y1.len)
y2.len <- unlist(y2.len)
table(y1.len, y2.len) #5 es el mayor
library(kernlab)
# consideramos 4 familias de kernels, con mayores esperanzas en el spectrum de longitud fija y lambda=1
#encontramos de mayor utilidad los kernels normalizados
spectrum.kernel <- stringdot(type="spectrum",length=3,
                             normalized=TRUE, lambda=1)
constante.kernel <- stringdot(type='constant', normalized = FALSE)
boundrange.kernel <- stringdot(type ='boundrange', length = 10,
                               normalized = TRUE, lambda = 1)
exponential.kernel <- stringdot(type='exponential',
                                lambda = .5, normalized = TRUE)

k <- spectrum.kernel
#construimos la primer matriz de similaridades, con todos los datos del archivo
K.Gram <- kernelMatrix(kernel=k ,
                       x=strings$description_x,
                       y=strings$description_y)
#realizamos Kernel - pca considerando un kernel gaussiano con parametroigual a la
#unidad, porque la matriz de similaridades esta centrada
kernel.pca <- kpca(K.Gram, kernel = "rbfdot", kpar = list(sigma = 1))
# revisamos la varianza explicada por las 10 primeras componentes principales
val.pro <- kernel.pca@eig
plot(cumsum(kernel.pca@eig)[1:10]/sum(kernel.pca@eig))
#guardamos los datos rotados: NOTA no tiene porque ser cuadrada la
# matriz de vectores propios, pues depende del rango de la matriz de
# similaridades
rotacion <- kernel.pca@rotated
rotacion <- as.data.frame(rotacion)
rotacion$etiqueta <- factor(strings$same_security)
library(ggplot2) #graficamos la proyeccion en las dos primeras componentes
ggplot(rotacion, aes(x =V1, y = V2, color=etiqueta, alpha=.1))+ geom_point()+
  ggtitle('Kernel: spectrum, n=3, lambda=1') + theme_minimal() + xlab('Primer kernel-componente principal')+
  ylab('Segunda kernel-componente principal')+ guides(alpha = FALSE)
###########inciso b
set.seed(0) #fijamos la semilla para comparar la pseudo muestra aleatoria
K.Gram.train <- kernelMatrix(kernel=k , #construimos la matriz de similaridades
                       x=strings$description_x) #considerando solo las etiquetas 'description_x'
kernel.pca.train <- kpca(K.Gram.train, kernel = "rbfdot", kpar = list(sigma = 1)) #kernel pca igual que en el
      caso anterior
index <- sample(1:dim(strings)[1], dim(strings)[1]  ) #indice de la muestra aleatoria
componentes.prin.train <- kernel.pca.train@pcv #vectores propios de kernel-pca
test <- strings[index, ] #conjunto de test
K.Gram.test <- kernelMatrix(kernel=k ,y = strings$description_x,
                            x=test$description_y) #calculamos la similaridad entre el
             #conjunto de train y el de test
res <- predict(kernel.pca.train, K.Gram.test) #rotamos el conjunto test
    #se procede a calculo explicito de las distancias de cada observacion
    #en el conjunto test contra todas las observaciones rotadas en el conjunto de train
foo <- matrix(rep(-1, dim(res)[1]*dim(strings)[1]), ncol = dim(strings)[1])
```

```
for (i in 1:dim(foo)[1])
{
  for(j in 1:dim(foo)[2])
  {
    foo[i, j] <- sum((res[i,]-kernel.pca.train@rotated[j,])**2)**.5
  }
}
mas.proximo <- apply(foo, 1, which.min ) #determinamos la observacion mas cercana del conjunto train
train <- kernel.pca.train@rotated[,1:2]
train <- as.data.frame(train)
test$res <- strings$description_x[mas.proximo] #agregamos al test para comparar
test$resul <- test$description_x == test$res #calificamos prediccion vs etiqueta
round(table(test$same_security,test$resul)/sum(table(test$same_security,test$resul)),2) #resumen
table(strings$same_security)/dim(strings)[1]
res <- as.data.frame(res)
y_hat <- train[mas.proximo,]
vis <- cbind(y_hat, res)
colores <- c('purple', 'black', 'orange', 'red', 'turquoise', 'blue4',
             'green4', 'salmon')
vis$index <- rep(colores, 3)
ggplot(train, aes(x =V1, y = V2, color=I('lightblue'), alpha=1))+ geom_point()+
  ggtitle('Train') + theme_minimal() + xlab('Primer kernel-componente principal')+
  ylab('Segunda kernel-componente principal')+ guides(alpha = FALSE) +
  geom_point(data=res[, 1:2], aes(x =V1, y = V2,
                                  color=(vis$index), alpha =1))+
  geom_point(data=y_hat[, 1:2], aes(x =V1, y = V2,
                                    color=(vis$index), alpha =1))+
  xlim(c(-26, 13))+ylim(c(-20,15))
```