Tema TFM

Clasificación de alimentos de origen vegetal según sus macronutrientes mediante el Análisis de Cluster

Entrega 2

Grupo 2

Miembros del grupo

Infante Ortega, Lady Viviana.

Sánchez Fabre, Alex Antonio.

Rodriguez Muñoz, Ricardo.

Sánchez Fabre, Felipe Andrés.

17 de julio de 2023

Contenido

1. Establecimiento de Hipótesis

Actualmente existe una necesidad patente de brindar a los consumidores veganos una guía más precisa y efectiva para la selección de alimentos que se ajusten a sus necesidades nutricionales. Aunque la dieta vegana ha demostrado ser beneficiosa en múltiples aspectos, su correcta ejecución puede presentar varios desafíos, por ejemplo la ausencia de una orientación apropiada conlleva que los consumidores pueden tener dificultades para mantener una dieta vegana saludable y equilibrada. Cabe mencionar que la elección manual de alimentos puede llevar a errores o inconsistencias, lo cual puede generar confusiones y desinformación para los consumidores, ya que la selección de alimentos según su contenido nutricional suele ser un proceso complejo por la cantidad de variables involucradas y las diferencias nutricionales sutiles, pero significativas entre los alimentos pueden hacer que la correcta elección manual de alimentos sea un gran desafío.

Hipótesis: La aplicación de algoritmos de aprendizaje no supervisado pueden identificar y agrupar de manera coherente los alimentos veganos, tales como soja, seitán y tofu, en diferentes grupos o clusters en base a la similitud de su composición nutricional. Basándose en similitudes y diferencias en la composición de sus macronutrientes y eventualmente, pero en menor grado de alguno de sus micronutrientes más relevantes. Al lograr una agrupación efectiva, se abre una posibilidad de brindar a los consumidores de productos veganos, profesionales del área de nutrición y otras partes interesadas de un medio de elección más informado, permitiendo la selección de aquellos alimentos que satisfagan de mejor manera sus requerimientos nutricionales individuales. En este sentido, se espera que los clusters nutricionales establecidos denoten variaciones notorias en cuanto a sus perfiles de macronutrientes, como las proteínas, las grasas y los carbohidratos entre los distintos tipos de alimentos veganos considerados en este trabajo.

La comprobación de esta hipótesis considera la recolección de datos nutricionales de alimentos veganos y la posterior aplicación de algoritmos de agrupación, como K-means, DBSCAN, o agrupación jerárquica. La eficacia de la agrupación se podría evaluar considerando tanto la cohesión dentro de los clusters, en términos de la similitud nutricional de los alimentos agrupados, como la separación entre los clusters, refiriéndonos a las diferencias nutricionales entre éstos.

Por lo tanto, este trabajo aborda el problema de cómo simplificar y mejorar el proceso de selección nutricional de los alimentos veganos, con el objetivo de proporcionar a los consumidores una mejor orientación nutricional. Al utilizar algoritmos de agrupación no supervisados, se busca identificar patrones en los datos nutricionales y agrupar los alimentos en grupos basados en su composición nutricional. De esta manera, los consumidores veganos podrán tomar decisiones más informadas sobre su dieta y tomar decisiones sobre productos específicos que respondan a sus intereses.

2. Adecuación de las fuentes.

Al partir un proyecto de datos generalmente iniciamos con una exploración de los mismos, que para nuestro caso es la base de datos de "Open Food Facts"; esta exploración implicó una meticulosa etapa de limpieza y validación de datos. Esta fase inicial incluyó varias transformaciones importantes. En primer lugar, realizamos una serie de transformaciones en los nombres de los productos que decidimos estudiar: Soja, Tofu y Seitán. Convertimos la columna “product_name” a minúsculas, eliminamos los acentos y normalizamos los nombres. Ajustamos además el tipo de dato de dicha columna a “StringType”, una necesidad de nuestro entorno de trabajo, “PySpark”.

Procedimos a identificar el idioma de los productos en la columna “product_name”. En donde descubrimos una diversidad de idiomas en los que se presentaban los productos, entre los cuales encontramos inglés, portugués, alemán, catalán, vietnamita, tagalo, turco, indonesio, somalí, suajili, croata, húngaro, galés, noruego, francés, español, sueco, polaco, rumano, lituano y persa. Posteriormente, tradujimos los nombres de los productos a todos los idiomas antes mencionados para poder realizar una búsqueda exhaustiva en la base de datos. Con la lista de nombres de productos traducida, procedimos a filtrar la base de datos en función de esta. A continuación, generamos una lista de exclusión de alimentos para refinar aún más nuestros resultados. Esta lista también fue traducida a los idiomas pertinentes.

Con el conjunto de datos filtrado y listo para el análisis, implementamos varias técnicas para garantizar su calidad. Estas técnicas incluyen:

1. Conversión de todos los datos a mayúsculas: Este paso garantiza la uniformidad en el conjunto de datos y facilita su análisis al eliminar las diferencias causadas por las variaciones en las mayúsculas y minúsculas.
2. Eliminación de filas con todos los valores nulos: Estas filas son irrelevantes para nuestro análisis ya que no aportan información útil.
3. Tratamiento de valores “NaN”: Reemplazamos los valores “NaN” por "vacío" (‘’) en el caso de las variables de cadena (StringType) y 0 en el caso de las variables numéricas. Este paso ayuda a mantener la coherencia de los datos y evita posibles errores durante el análisis.
4. Eliminación de datos duplicados: Para evitar la distorsión de nuestro análisis debido a la repetición, eliminamos todos los datos duplicados.
5. Estandarización de los nombres de las columnas: Este paso implica la eliminación de caracteres especiales y tildes, y el reemplazo de espacios por guiones bajos. Esto facilita el manejo de los datos y evita posibles errores durante el análisis.

Además, descartamos las columnas numéricas cuya suma total era igual a 0, ya que estas columnas no aportarían ningún valor a nuestros análisis futuros. Posteriormente, normalizamos los nombres de los países y añadimos el código “ISO3” para cada uno de ellos usando la librería “pycountry”. Asimismo, incorporamos la columna “PIB” o Producto Interno Bruto (en US$ a precios actuales de 2020) para cada país, con información rescatada directamente desde el Banco Mundial. Como un paso adicional, transformamos las variables categóricas a numéricas, a excepción de las columnas de nombre de producto, nombre de país e “ISO3”.

Finalmente, dividimos nuestro conjunto de datos en tres partes, una para cada producto de interés (Seitán, Tofu y Soja), utilizando los nombres de los productos en los diferentes idiomas identificados. Como resultado, obtuvimos cuatro conjuntos de datos: uno general que incluye todos los productos (Seitán, Tofu, Soja), y uno específico para cada producto.

En este punto logramos transformar el dataset de “Open Food Facts” en un recurso analítico robusto y bien definido. Hemos tomado medidas cuidadosas para garantizar la consistencia, la relevancia y la precisión de nuestros datos. Sin embargo, este proceso inicial es solo la preparación para las etapas de análisis más intensivas que están por venir. La laboriosa tarea de limpieza y validación de los datos proporcionará una base sólida para los modelos predictivos y los análisis estadísticos.

Nuestro enfoque minucioso al identificar los idiomas de los productos y realizar traducciones precisas para cada uno de ellos, nos permitirá tener una cobertura global al explorar los datos. Esto es especialmente importante dada la diversidad inherente de la base de datos "Open Food Facts", que recopila información de productos alimenticios de todo el mundo.

Además, el proceso de categorización y exclusión de alimentos nos permite mantener la integridad temática de nuestro estudio. Al concentrarnos en Seitán, Tofu y Soja, y al mismo tiempo excluir productos que, aunque puedan contener estos ingredientes, no son representativos de ellos, estamos garantizando que nuestro análisis sea lo más específico y relevante posible. Al combinar esta precisión temática con nuestras prácticas de limpieza y validación de datos, estamos en una posición ideal para producir análisis significativos y precisos en las etapas posteriores de nuestro estudio.

El código fuente de este proceso, lo pueden revisar en el siguiente repositorio de Github:

- https://github.com/atilianno/tfm_master_ds/blob/main/1%20-%20BackEnd/Python/Datos_TFM_Master_DS%20-%20v0.5.ipynb

3. Hitos temporales de trabajo (Timing)

Abordaremos el desarrollo de este trabajo utilizando la metodología SCRUM, estructurada en Sprints con hitos establecidos que permitan medir el avance del trabajo.

Los hitos respresentan las entregas al tutor, mientras que las tareas definidas para ejecutar durante cada Sprint quedan plasmadas en el backlog, dichas tareas estan planteadas en forma de issues con los plazos y responsables. Cada sprint tiene una duración aproximada de 2 semanas. Al final del sprint, se hace una reunión con los miembros del projecto para mostrar el avance obtenido en ese lapso.

Sprint 1 - Preparación y entendimiento del conjunto de datos:

Durante este primer sprint, se recogieron y se prepararon los datos. Esta etapa involucró la recolección de un conjunto de datos exhaustivo y relevante sobre alimentos veganos, específicamente soya texturizada, tofu y seitán, con un enfoque en macronutrientes y micronutrientes. Además, se realizó un análisis exploratorio de los datos para comprender sus características, identificar desafíos y plantear las preguntas de investigación pertinentes.

Sprint 2 - Preprocesamiento de datos y selección de características:

En este sprint, se pre-procesan los datos y se seleccionan las características relevantes para la generación de clusters. Se implementaron técnicas de limpieza de datos y se seleccionaron las características nutricionales que se utilizaron en los algoritmos de agrupación.

Sprint 3 - Implementación de algoritmos de agrupación:

Se implementaron diversos algoritmos, entre ellos detección de outliers con el algoritmo Local Outlier Factor (LOF) y de algoritmos agrupación no supervisados abordados en el módulo 5, tales como k-means y agrupación jerárquica. En esta etapa se aplicaron los algoritmos más apropiados, se ajustaron sus parámetros y se generaron los clusters iniciales de alimentos veganos.

Sprint 4 - Evaluación y afinamiento de clusters:

Se evaluarán y ajustarán los clusters. Se implementarán medidas para evaluar la calidad de los clusters, y se ajustarán los parámetros de los algoritmos para mejorar la coherencia y utilidad de los clusters.

Sprint 5 - Interpretación y presentación de resultados:

Se interpretarán y presentarán los resultados. Se interpretarán los clusters en términos nutricionales, y se desarrollarán visualizaciones y reportes para comunicar los hallazgos de manera clara y comprensible.

Sprint 6 - Desarrollo de un sistema informático para el acceso de los resultados a los consumidores

Para garantizar que los resultados de nuestro trabajo sean accesibles y útiles para los consumidores y otras partes interesadas, se diseñará y desarrollará un sistema informático. Este sistema permitirá a los usuarios explorar los grupos de alimentos y entender sus perfiles nutricionales de manera clara y concisa. El desarrollo de este sistema incluirá las etapas de diseño, implementación del sistema, prueba del sistema y despliegue del sistema.

Sprint 7 - Redacción de la memoria del proyecto

Elaboración de la memoria del trabajo, un documento exhaustivo que se ceñirá a las exigencias establecidas por la Universidad. Esta memoria detalla los métodos utilizados, los hallazgos obtenidos, las decisiones tomadas y las lecciones aprendidas a lo largo del trabajo.

Cada uno de estos sprints está precedido por una reunión de planificación, seguida por un sprint de trabajo, y finalizando con una revisión y retrospectiva del sprint para evaluar el progreso y adaptar el plan de trabajo según sea necesario.

El seguimiento del proyecto, lo pueden revisar en el siguiente repositorio de Github:

- https://github.com/users/atilianno/projects/2/views/1?layout=board

4. Estimación de los costes del proyecto.

ACTIVIDAD	CATEGORÍA	TAREA	COSTE	UNIDAD	DESCRIPCIÓN
Desarrollo de software	CAPEX	Análisis y relevamiento de requerimientos	250	USD	Tiempo y recursos necesarios para comprender los objetivos del proyecto, identificar los requisitos funcionales y no funcionales, y documentar las necesidades de los potenciales clientes
Desarrollo de software	CAPEX	Diseño de la arquitectura	250	USD	Tiempo y recursos necesarios para establecer la definición de la estructura y la organización general de la aplicación web, la elección de tecnologías y herramientas adecuadas, y la planificación de la arquitectura de base de datos.
Desarrollo de software	CAPEX	Diseño de la interfaz de usuario (UI/UX)	500	USD	Tiempo y recursos necesarios para la creación de prototipos, la definición de la experiencia de usuario, el diseño visual de la interfaz y la creación de elementos gráficos necesarios.
Desarrollo de software	CAPEX	Desarrollo de funcionalidades	1000	USD	Tiempo y recursos necesarios para la implementación de las diferentes características y funcionalidades de la aplicación web, como por ejemplo el registro de usuarios, la gestión de las transacciones, la generación de informes, etc.
Desarrollo de software	CAPEX	Pruebas de software	250	USD	Tiempo y recursos necesarios para realizar pruebas de calidad, tanto funcionales como no funcionales, para garantizar que la aplicación web cumpla con los requerimientos establecidos y funcione correctamente en diferentes escenarios
Desarrollo de software	CAPEX	Documentación	500	USD	Tiempo y recursos para la creación de documentación técnica y de usuario, que describe el funcionamiento de la aplicación, los requisitos del sistema, los procedimientos de instalación y uso, entre otros

Desarrollo de software	CAPEX	Implementación y despliegue	250	USD	Tiempo y recursos necesarios para configurar la infraestructura de alojamiento, instalar y configurar la aplicación en el entorno de producción, y asegurar que esté lista para su uso
Desarrollo de software	CAPEX	Capacitación y soporte inicial			No aplica
SubTotal 'Desarrollo de software - CAPEX'			**3000**	**USD**	
Diseño y experiencia de usuario	CAPEX	Investigación de usuarios y análisis de necesidades.	200	USD	Tiempo y recursos necesarios para comprender a los usuarios objetivo de la aplicación y sus necesidades. Esto puede incluir la realización de entrevistas, encuestas, análisis de datos y la investigación de la competencia
Diseño y experiencia de usuario	CAPEX	Creación de personas y escenarios			No aplica
Diseño y experiencia de usuario	CAPEX	Diseño de arquitectura de información	500	USD	Tiempo y recursos para la organización y estructuración de la información dentro de la aplicación web transaccional. Se crean mapas de sitio, diagramas de flujo y jerarquías de navegación para garantizar que los usuarios puedan encontrar y acceder a la información de manera intuitiva.
Diseño y experiencia de usuario	CAPEX	Creación de prototipos	200	USD	Tiempo y recursos para la creación de prototipos interactivos de la interfaz de usuario de la aplicación web. Estos prototipos permiten probar y validar la usabilidad y la experiencia del usuario antes de pasar a la etapa de desarrollo
Diseño y experiencia de usuario	CAPEX	Diseño visual	150	USD	Tiempo y recursos para el diseño visual de la interfaz de usuario de la aplicación web transaccional. Se crean los elementos gráficos, como logotipos, iconos, botones, tipografías, esquemas de color, para asegurar una apariencia coherente y atractiva.

Diseño y experiencia de usuario	CAPEX	Pruebas de usabilidad	250	USD	Implica llevar a cabo pruebas con usuarios reales para evaluar la usabilidad de la aplicación web transaccional y obtener retroalimentación sobre la experiencia del usuario
Diseño y experiencia de usuario	CAPEX	Iteraciones y refinamiento.	100	USD	Tiempo y recursos necesarios para realizar iteraciones y refinamientos en el diseño en función de los comentarios de los usuarios y los resultados de las pruebas de usabilidad. Esto asegura que la interfaz y la experiencia del usuario sean óptimas
SubTotal 'Diseño y experiencia de usuario - CAPEX'			**1400**	**USD**	
Infraestructura y alojamiento	CAPEX	Selección de la infraestructura	100	USD	Evaluar y seleccionar el tipo de infraestructura necesaria para alojar la aplicación web. Se debe optar entre las opciones disponibles, es decir, entre servidores dedicados, servidores virtuales, servicios en la nube (como Amazon Web Services, Google Cloud Platform, Microsoft Azure),.
Infraestructura y alojamiento	CAPEX	Configuración de la infraestructura	250	USD	Tiempo y recursos necesarios para configurar y establecer la infraestructura de alojamiento. Esto implica la instalación y configuración del sistema operativo, configuración de redes, seguridad, firewalls, etc.
Infraestructura y alojamiento	CAPEX	Escalabilidad			No aplica
Infraestructura y alojamiento	CAPEX	Seguridad			No aplica
SubTotal 'Infraestructura y alojamiento - CAPEX'			**350**	**USD**	

Infraestructura y alojamiento	OPEX	Costos de servidores y equipos	50	USD/Mes	Costo de adquisición o alquiler de los servidores físicos, o el costo asociado al uso de servidores virtuales en la nube. Se deben tener en cuenta los costos de otros equipos o componentes necesarios, como switches, enrutadores, cables, entre otros.
Infraestructura y alojamiento	OPEX	Licencias de software	25	USD/Mes	Adquisición de licencias de software para la infraestructura (por ejemplo, sistemas operativos, bases de datos, servidores web),
Infraestructura y alojamiento	OPEX	Gestión y monitorización de la infraestructura	200	USD/Mes	Considera los costos relacionados con la gestión y monitorización de la infraestructura. Esto puede incluir el monitoreo del rendimiento de los servidores, la gestión de actualizaciones de software, la implementación de medidas de seguridad, la administración de copias de seguridad, entre otros
Infraestructura y alojamiento	OPEX	Ancho de banda y tráfico	25	USD/Mes	Costos asociados al ancho de banda y tráfico que la aplicación web transaccional requerirá. Esto puede incluir tarifas de servicio de Internet, servicios de CDN (Content Delivery Network) para optimizar la entrega de contenido, o planes de hosting que incluyan una cantidad de ancho de banda específica.
SubTotal 'Infraestructura y alojamiento - OPEX'			**300**	**USD/Mes**	
Mantenimiento y soporte	OPEX	Corrección de errores y problemas técnicos	10	USD/Mes	Tiempo y recursos necesarios para corregir cualquier error o problema técnico que pueda surgir en la aplicación web transaccional después de su lanzamiento. Esto puede incluir la identificación de errores, el desarrollo de soluciones y la implementación de parches o actualizaciones.
Mantenimiento y soporte	OPEX	Actualizaciones de software y tecnología	10	USD/Mes	Costos asociados con las actualizaciones regulares del software y las tecnologías utilizadas en la aplicación web transaccional. Esto puede incluir actualizaciones de seguridad, mejoras de rendimiento, nuevas funcionalidades o adaptaciones a cambios en los estándares o regulaciones.

Mantenimiento y soporte	OPEX	Monitoreo y rendimiento	10	USD/Mes	Monitoreo constante del rendimiento de la aplicación web transaccional para garantizar su correcto funcionamiento y alta disponibilidad. Esto puede implicar el uso de herramientas de monitoreo, análisis de registros y métricas, y la identificación y solución de problemas de rendimiento
Mantenimiento y soporte	OPEX	Copias de seguridad y recuperación de datos	30	USD/Mes	Costos asociados con la realización regular de copias de seguridad de los datos de la aplicación web transaccional y la implementación de mecanismos de recuperación en caso de pérdida de datos o fallas del sistema.
Mantenimiento y soporte	OPEX	Actualizaciones de contenido	10	USD/Mes	Costos asociados con la actualización regular del contenido de la aplicación web transaccional, como información de productos, promociones, términos y condiciones, entre otros. Esto puede requerir la implementación de un sistema de gestión de contenido (CMS) o la intervención de desarrolladores para realizar las actualizaciones
Mantenimiento y soporte	OPEX	Soporte técnico y atención al cliente	30	USD/Mes	Costos asociados con la provisión de soporte técnico y atención al cliente para los usuarios de la aplicación web transaccional. Esto puede implicar la asignación de personal de soporte, la implementación de sistemas de tickets o chat en vivo, y la respuesta a consultas, incidencias o solicitudes de los usuarios.
Mantenimiento y soporte	OPEX	Mejoras y optimizaciones	50	USD/Mes	Costos asociados con la implementación de mejoras y optimizaciones en la aplicación web transaccional para garantizar su rendimiento, usabilidad y escalabilidad a largo plazo. Esto puede incluir la optimización de consultas de base de datos, la mejora de la velocidad de carga, la incorporación de nuevas funcionalidades, entre otros.
SubTotal 'Mantenimiento y soporte - OPEX'			**150**	**USD/Mes**	

Marketing y promoción	CAPEX	Investigación de mercado y análisis de competencia	200	USD	Tiempo y recursos necesarios para investigar el mercado objetivo de la aplicación web transaccional y analizar a la competencia existente. Esto puede incluir estudios de mercado, análisis de tendencias, identificación de segmentos de mercado y evaluación de la competencia directa e indirecta.
Marketing y promoción	CAPEX	Estrategia de marketing y planificación	300	USD	Costos asociados con la definición de una estrategia de marketing adecuada para la promoción de la aplicación web transaccional. Esto incluye la planificación de acciones y tácticas de marketing, la definición de objetivos, el establecimiento de indicadores clave de rendimiento (KPIs) y la elaboración de un plan de ejecución
Marketing y promoción	CAPEX	Desarrollo de marca y diseño de identidad visual	500	USD	Costos asociados con la creación de la identidad de marca de la aplicación web transaccional. Esto implica el diseño del logotipo, la selección de colores y tipografías, el desarrollo de la identidad visual coherente con la propuesta de valor de la aplicación.
SubTotal 'Marketing y promoción - CAPEX'			**1000**	**USD**	
Marketing y promoción	OPEX	Desarrollo de materiales de marketing	10	USD/Mes	Costos asociados con la creación de materiales de marketing y promoción para la aplicación web transaccional. Esto puede incluir la producción de folletos, volantes, presentaciones, vídeos promocionales, imágenes para redes sociales, entre otros
Marketing y promoción	OPEX	Estrategias de publicidad en línea	10	USD/Mes	Costos asociados con la implementación de estrategias de publicidad en línea para promocionar la aplicación web transaccional. Esto puede incluir campañas de publicidad en motores de búsqueda (SEM), publicidad en redes sociales, anuncios gráficos en sitios web relevantes, entre otros.

Marketing y promoción	OPEX	Estrategias de marketing de contenidos	10	USD/Mes	Costos asociados con la creación y promoción de contenido relevante y de calidad para atraer a los usuarios y generar interés en la aplicación web transaccional. Esto puede incluir la creación de blogs, artículos, guías, videos educativos, infografías, entre otros.
Marketing y promoción	OPEX	Estrategias de SEO (optimización para motores de búsqueda)	40	USD/Mes	Costos asociados con la optimización de la aplicación web transaccional para mejorar su visibilidad en los resultados de búsqueda orgánica. Esto puede implicar la realización de investigaciones de palabras clave, la optimización del contenido, la mejora de la estructura del sitio web y la obtención de enlaces relevantes
Marketing y promoción	OPEX	Estrategias de relaciones públicas y relaciones con la comunidad	40	USD/Mes	Costos asociados con la implementación de estrategias de relaciones públicas y relaciones con la comunidad para promover la aplicación web transaccional. Esto puede incluir la participación en eventos, la colaboración con influencers, la generación de relaciones con la prensa y la participación en comunidades en línea.
SubTotal 'Marketing y promoción - OPEX'			**110**	**USD/Mes**	
Seguridad	CAPEX	Análisis de riesgos y evaluación de amenazas	250	USD	Implica realizar un análisis exhaustivo de los posibles riesgos y amenazas que pueden afectar la seguridad de la aplicación web transaccional. Esto incluye identificar vulnerabilidades potenciales y evaluar el impacto que podrían tener en la aplicación y los datos
Seguridad	CAPEX	Diseño e implementación de medidas de seguridad	250	USD	Costos asociados con el diseño e implementación de medidas de seguridad adecuadas para proteger la aplicación web transaccional. Esto puede incluir la implementación de firewalls, sistemas de detección y prevención de intrusiones, autenticación de usuarios, control de acceso, encriptación de datos, entre otros

Seguridad	CAPEX	Pruebas de seguridad	500	USD	Implica realizar pruebas de seguridad para identificar y resolver vulnerabilidades antes de lanzar la aplicación web transaccional. Esto puede incluir pruebas de penetración, pruebas de seguridad automatizadas, auditorías de seguridad y revisiones de código
Seguridad	CAPEX	Certificados SSL y encriptación	500	USD	Costos asociados con la adquisición y configuración de certificados SSL para establecer una conexión segura entre los usuarios y la aplicación web. Además, se debe considerar la encriptación de datos sensibles, tanto en reposo como en tránsito
Seguridad	CAPEX	Capacitación y concientización en seguridad			No aplica
Seguridad	CAPEX	Cumplimiento de normativas y regulaciones	250	USD	Si la aplicación web transaccional está sujeta a normativas y regulaciones específicas, como el Reglamento General de Protección de Datos (GDPR) en Europa o las regulaciones de la industria de pagos, se deben considerar los costos asociados con el cumplimiento de estas normativas
SubTotal 'Seguridad - CAPEX'			**1750**	**USD**	
Seguridad	OPEX	Actualizaciones y parches de seguridad	20	USD/Mes	Costos relacionados con la aplicación de actualizaciones y parches de seguridad para mantener la aplicación web transaccional protegida contra nuevas amenazas y vulnerabilidades que puedan surgir con el tiempo
Seguridad	OPEX	Monitoreo y gestión de seguridad	20	USD/Mes	Costos asociados con el monitoreo constante de la seguridad de la aplicación web transaccional. Esto puede incluir la implementación de herramientas de monitoreo de seguridad, la configuración de alertas y la asignación de recursos para responder a incidentes de seguridad.

SubTotal 'Seguridad - OPEX'			**40**	**USD/Mes**	
	SUBTOTAL	**CAPEX**	**7500**	**USD**	
	SUBTOTAL	**OPEX**	**600**	**USD/Mes**	

Uno de los 'entregables' del proyecto es un sitio web en donde los distintos stakeholders puedan consultar acerca de un determinado producto, en principio sólo podrá hacerlo sobre los productos seleccionados como alcance en el presente trabajo, es decir, Tofu, Seitán y Soya texturizada. La consulta le permitirá establecer a qué clúster pertenece el producto específico sobre el cual está haciendo la consulta. Supongamos que para el producto Tofu, se establecen que existen 3 clústers, siendo estos los siguientes;

Tofu – Clúster 1: Productos que son altos en proteínas, bajos en carbohidratos y bajos en grasas.

Tofu – Clúster 2: Productos que son bajos en proteínas, altos en carbohidratos y bajos en grasas.

Tofu – Clúster 3: Productos que son bajos en proteínas, bajos en carbohidratos y bajos en grasas.

El usuario o stakeholeder del sitio web, podrá entrar en la página, identificar el producto sobre el cual desea o necesita hacer la consulta, y recibirá como respuesta las características del clúster al cual pertenece, y además, la especificación de las características de los demás clúster asociados al producto consultado, en este caso, el tofu. Con esta información, el usuario podrá tomar la decisión de compra que le parezca pertinente, ya que tendrá disponible ante él una cantidad de información que podrá ser utilizada antes de tomar la mencionada decisión. Por ejemplo, si el usuario es un habitual consumidor de un tofu que pertenece al 'Tofu – Clúster 2', porque es la marca que venden en el comercio cercano de su domicilio, pero se percata que éste es 'alto en carbohidratos' y que existen otras marcas de tofu que no tienen esta característica, perfectamente podría decidir cambiar su hábito, y comenzar a adquirir una marca de alguno de los otros dos clústers.

Volviendo a la estimación de costes, se adjunta una planilla creada para tales fines. Las distintas 'Tareas' necesarias para materializar el proyecto se agruparon en 'Actividades', siendo una 'Actividad' un concepto más general que agrupa un determinado conjunto de "Tareas', estás últimas corresponden a labores muy específicas, las que además podrían ser clasificadas en dos grandes categorías; 'CAPEX' y 'OPEX'. El término 'CAPEX', por sus siglas en inglés ('Capital Expenditure'), corresponden a lo que en español denominamos habitualmente como 'Inversión', por lo que se trata de costes que se manifiestan sólo una vez, habitualmente en las etapas iniciales del proyecto. Por otro lado, el término 'OPEX', por sus siglas en inglés ('Operational Expenditure') , corresponden a los costes operacionales, los que tienen una recurrencia habitualmente mensual o anual, y cuando se da este segundo caso, es habitual que el valor se prorratee mensualmente, de modo que aunque no se manifieste directamente el gasto, los montos asociados se vayan provisionando y éstos estén disponibles, en el monto en que este costo operacional anual, se manifieste. Un caso representativo de este tipo de OPEX sería el pago de licencias de software anuales, las cuales perfectamente se pueden considerar un 'coste operacional' mensual, a pesar de que, en estricto rigor, no lo son.

Hechas estas precisiones, podemos indicar que nuestra estimación de costes nos arroja como grandes totales, un CAPEX de 7500 USD, y un OPEX de 600 USD/Mes. El ítem más alto está asociado a la confección misma de la página web, la que entre las actividades de 'Desarrollo de software', de 'Diseños y experiencia de usuario', de 'Infraestructura y alojamiento' y de 'Seguridad', requieran 6500 de los 7500 USD de CAPEX. Los restantes 1000 USD corresponden a tareas que se agrupan en la familia de actividades de 'Marketing y promoción'. Siguiendo con las tareas de 'Marketing y promoción', estas cuentan con un presupuesto anual de 1320 USD (110 USD/Mes), para aquellas que fueron categorizadas como OPEX y que se muestran en la planilla de costes. Este es el mismo caso de las actividades de 'Infraestructura y alojamiento', de

‘Mantenimiento y soporte’ y de ‘Seguridad’, las que cuentan con presupuestos mensuales de 300, 150 y 40 USD respectivamente.

Finalmente mencionar que en una primera etapa, el servicio será gratuito, por lo que sólo en caso que logre generar el interés todos los stakeholders, o al menos en particular de la comunidad vegana, se evaluaría la posibilidad real de que haya un cobro por el uso del servicio. Estimamos que, habiendo tantos servicios disponibles gratuitos hoy en la web, lo que genera en el ‘inconsciente colectivo’ la percepción que todos los servicios web deben ser gratuitos, es importante establecer si nuestro concepto tiene una demanda real, y sólo allí, preguntarse si habría una disposición a pagar por este de parte de los stakeholders.

5. Desarrollo de un modelo predictivo.

Para el desarrollo del modelo se ha decidido emplear la metodología combinada de K-means y PCA. Se ha implementado adicionalmente la metodología de Local Outlier Factor (LOF) para la detección de productos atípicos en el conjunto inicial de datos, así como también la revisión a nivel univariante. A continuación se presentan los resultados obtenidos para los productos Seitán, Tofu y Soja.

Como primera aproximación al análisis exploratorio de datos sobre los registros descargados (9424 filas por 78 columnas), se decidió realizar una segunda limpieza sobre el conjunto de entrada para excluir registros duplicados, vectores nulos y hacer un conteo de frecuencias del total de valores cero para cada variable, así como una representación gráfica del histograma en cada variable incluida en modelo y en cada Producto. Para ello, se identificó el porcentaje de valores faltantes por celda en cada variable. Sobre la base de los objetivos planteados, se decidió utilizar las siguientes variables cuantitativas continuas para el modelo: Energía, proteínas, grasas y carbohidratos por cada 100 gramos (**ENERGY_100G, PROTEINS_100G, FAT_100G y CARBOHYDRATES_100G**), la columna energía junto con los 3 macronutrientes de interés. Se ha explorado incluir las variables azúcar, sal, grasas saturadas y fibras, sin embargo, luego de revisar el contenido de estas, se decidió no contemplarlas para esta entrega por las siguientes razones: i) la sal y la fibra no forman parte de los objetivos de este estudio, ii) varias de estas están contenidas en las variables principales elegidas como los carbohidratos y grasas saturadas iii) se encuentran nulas en muchas observaciones y finalmente iv) muchas de ellas no tienen presencia de ciertos componentes en los productos examinados, a continuación se presenta el porcentaje de celdas vacías para estas variables por producto:

Variables Seitán	% de celdas nulas
PROTEINS_100G	4
ENERGY_100G	4
CARBOHYDRATES_100G	5
FAT_100G	5
ENERGY_KCAL_100G	6
SALT_100G	10
SODIUM_100G	10
SATURATED_FAT_100G	14
SUGARS_100G	15
NUTRISCORE_GRADE	48
NUTRISCORE_SCORE	53
NUTRITION_SCORE_FR_100G	53
ECOSCORE_SCORE	54
FIBER_100G	61
Resto de variables (60)	entre 92 y 100

Variables Tofu	% de celdas nulas
PROTEINS_100G	6
ENERGY_100G	6
FAT_100G	6
CARBOHYDRATES_100G	8
ENERGY_KCAL_100G	9
SATURATED_FAT_100G	15
SUGARS_100G	26
SALT_100G	27
SODIUM_100G	27
NUTRISCORE_GRADE	52
NUTRISCORE_SCORE	56
NUTRITION_SCORE_FR_100G	56
ECOSCORE_SCORE	58
FIBER_100G	59
Resto de variables (59)	entre 92 y 100

Variables Soja	% de celdas nulas
ENERGY_100G	10
PROTEINS_100G	10
CARBOHYDRATES_100G	11
FAT_100G	13
ENERGY_KCAL_100G	15
SATURATED_FAT_100G	19
SUGARS_100G	19
SODIUM_100G	29
SALT_100G	29
NUTRISCORE_GRADE	55
FIBER_100G	60
NUTRITION_SCORE_FR_100G	61
NUTRISCORE_SCORE	61
ECOSCORE_SCORE	69
ENERGY_KJ_100G	80
FRUITS_VEGETABLES_NUTS_ESTIN	86
Resto de variables (57)	entre 96 y 100

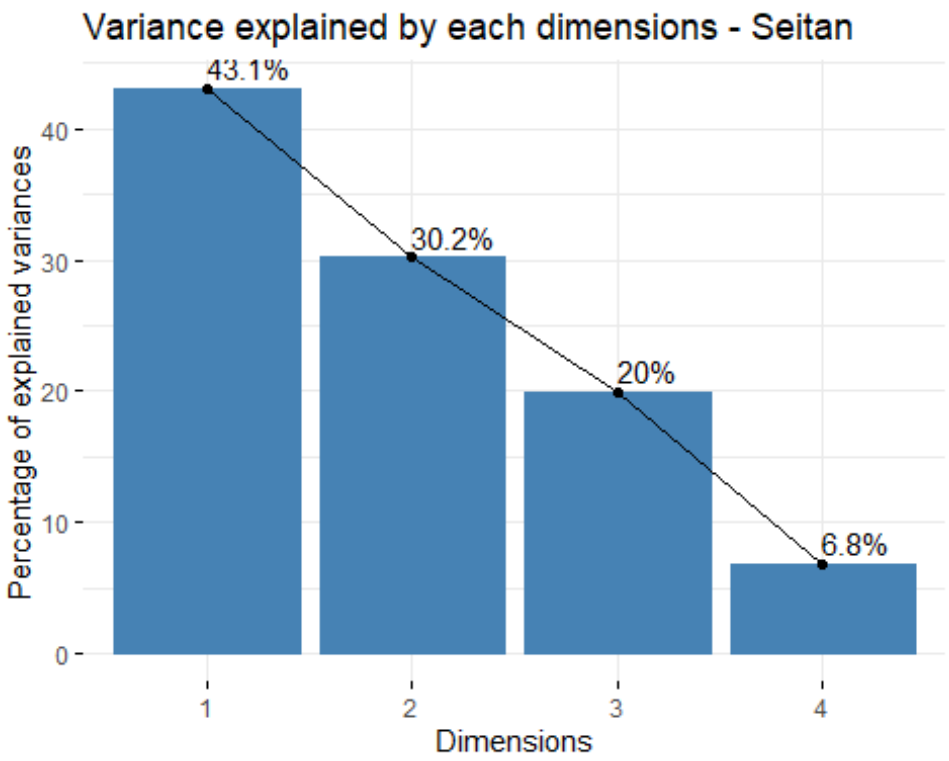
Adicionalmente, se identificó en el conjunto de entrada productos que no son objeto de este estudio, para ello, luego de hacer la consulta a la API, se han listado palabras que serán utilizadas para excluir esos productos utilizando el criterio de filtros negativos. Todavía la exclusión de estos registros es una actividad que requiere atención exhaustiva, dado que "contaminan" el conjunto de entrada y repercuten significativamente en el cálculo de las distancias, y por lo tanto en la formación de cluster mutuamente excluyentes.

Todos los resultados descritos a continuación responden a cálculos propios.

01. Seitan

Sobre el conjunto de datos clasificados como Seitán se ha realizado una Análisis de Componentes Principales (PCA), se observa como la proporción de varianza explicada es de 73% con los dos primeros autovalores elegidos.

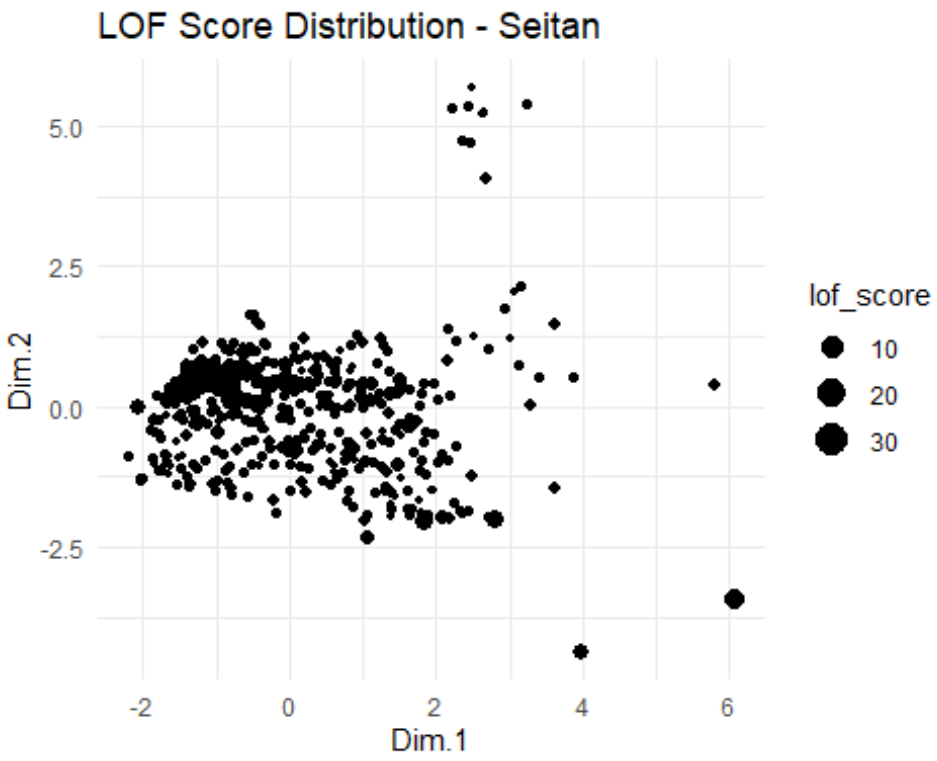
```
fviz_eig(seitan_pca, ncp = 6, addlabels = T, main = "Variance explained by
each dimensions - Seitan")
```



Ahora bien, teniendo en consideración el análisis exploratorio realizado previamente, se conoce que la distribución por variable tiene varios valores extremos. Para identificar estos registros se utilizará la Técnica multivariante LOF, a continuación los resultados:

Técnica LOF - Seitán

```
library(ggthemes)
seitan_a <- data.frame(seitan_pca$ind$coord[,1:3])
seitan_b <- cbind(seitan_a, lof_score = seitan_lof)
#seitan_b <- cbind(seitan_a, fraud = seitan_clean$, lof_score =
seitan_clean$lof)

seitan_lof_visual <- ggplot(seitan_b, aes(x=Dim.1 ,y=Dim.2)) +
    geom_point(aes(size=lof_score)) +
  ggtitle("LOF Score Distribution - Seitan")+
    theme_minimal()

seitan_lof_visual
```



La regla general de la puntuación LOF generaliza que si la puntuación LOF es superior a 1, es probable que sea un valor atípico. De alguna manera, se puede ajustar un umbral con la distribución de los datos. Veamos primero las estadísticas de la puntuación LOF.

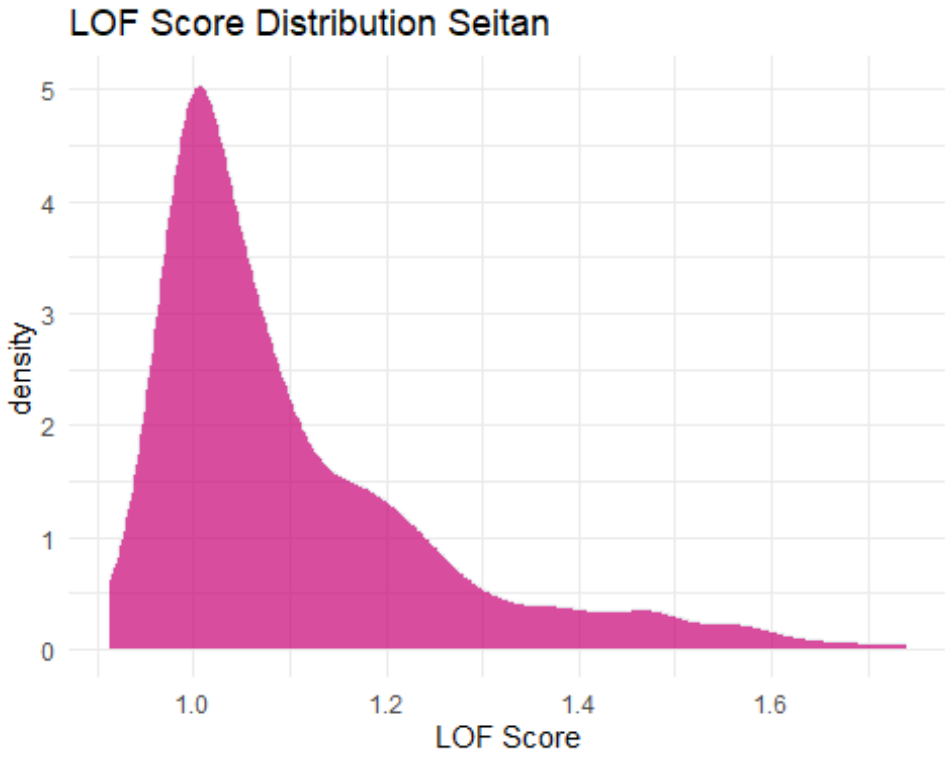
```
summary(seitan_b)
```

```
##      Dim.1              Dim.2              Dim.3            lof_score
##  Min.   :-2.2051   Min.   :-4.37419   Min.   :-8.2367   Min.   :0.9126
##  1st Qu.:-0.9834   1st Qu.:-0.68423   1st Qu.:-0.3688   1st Qu.:1.0014
##  Median :-0.3252   Median : 0.06254   Median :-0.0602   Median :1.0707
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   :   Inf
```

```
##  3rd Qu.: 0.8360   3rd Qu.: 0.55616   3rd Qu.: 0.3935   3rd Qu.:1.2404
##  Max.   : 6.0740   Max.   : 5.68950   Max.   : 5.4533   Max.   :   Inf
```

```
seitan_b %>%
  filter(lof_score <= 1.75) %>%
  ggplot( aes(x=lof_score)) +
    geom_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +
    scale_fill_manual(values="#8fce00") +
    xlab("LOF Score")+
  ggtitle("LOF Score Distribution Seitan")+
    theme_minimal() +
    labs(fill="")
```

La puntuación LOF tiene una puntuación máxima de más de 6. Si incluimos un punto que está lejos de la distribución, será difícil de visualizar. Por lo tanto, estableceremos la puntuación LOF hasta 1,75 y veremos la distribución de la puntuación LOF.



Vemos arriba, la puntuación LOF tiene muchos puntos con una puntuación superior a 1. Para clasificar un punto como atípico o no, podemos establecer el umbral más alto.

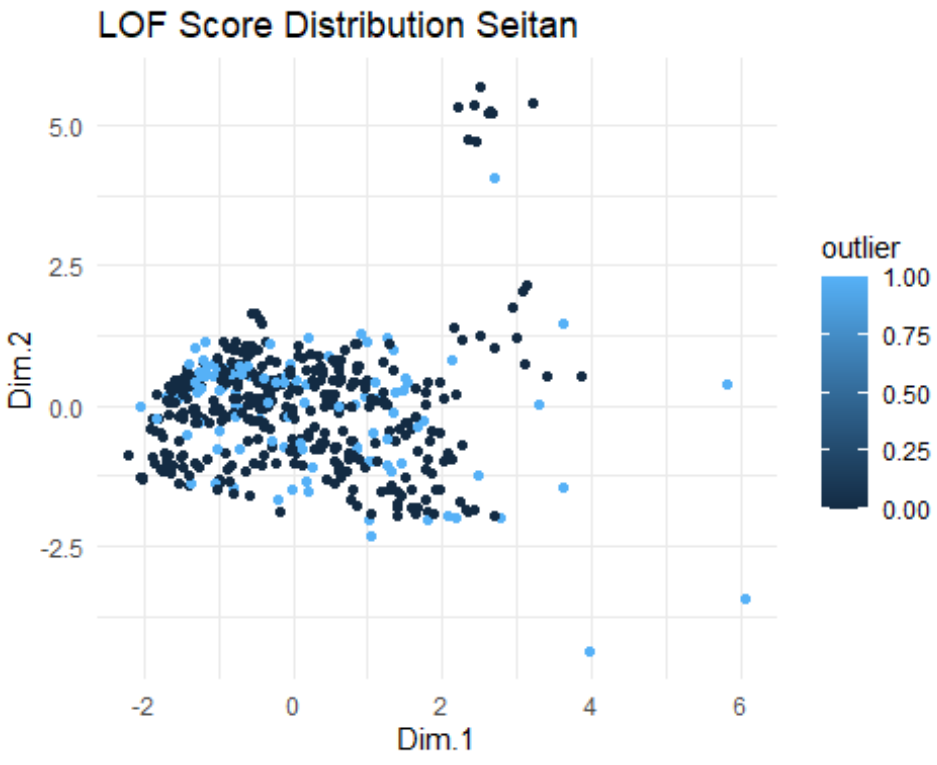
Un método para determinar el umbral es calcular el punto cuantil. Para este ejercició, se estableció un umbral del 80% como los puntos normales, mientras que el último 20 % se considerará como un valor atípico.

```
quantile(seitan_b$lof_score, probs = c(0, 0.8))
```

```
##        0%       80%
## 0.9125573 1.3319969
```

La proporción del 80 % de la puntuación LOF está por debajo de 1,3319969. Se utilizó este umbral para determinar si un punto cae por debajo del umbral; categorizamos ese punto como un valor atípico.

```
Una vez más, podemos visualizar la distribución del valor atípico para todas
las observaciones.



seitan_b <- seitan_b %>%
  mutate(outlier = ifelse(lof_score > 1.3319969, 1, 0))

seitan_lof_visual_b <- ggplot(seitan_b, aes(x=Dim.1 ,y=Dim.2, color=outlier))
+
    geom_point() +
  ggtitle("LOF Score Distribution Seitan")+
    theme_minimal()

seitan_lof_visual_b
```



La visualización anterior nos muestra que exiten varios productos atípicos para el conjunto del Seitán. Se requiere investigar estas observaciones de manera aislada. El alto puntaje de

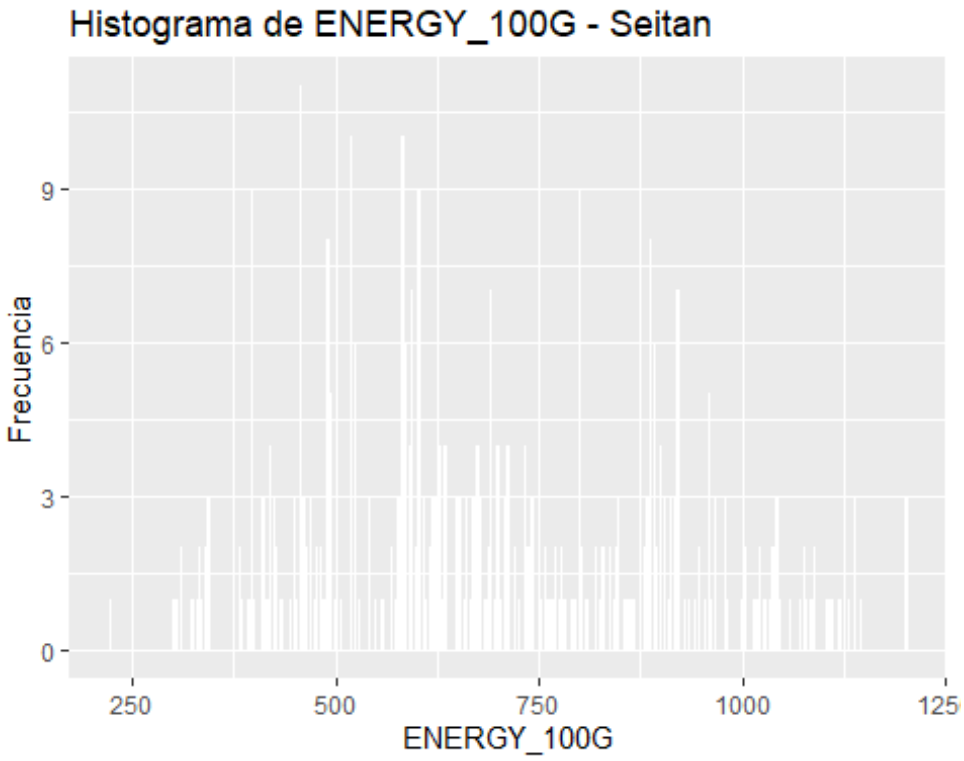
lof en los productos debe interpretarse con más criterio al momento de elegir los productos que son objeto de este estudio.

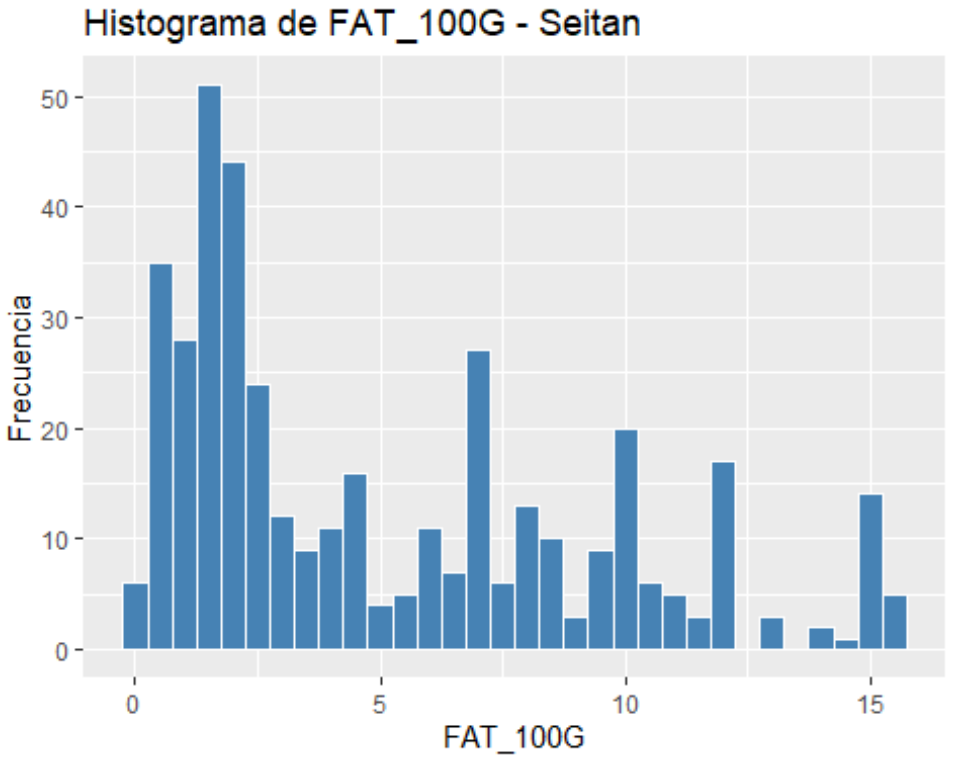
```
outliers <- seitan_b[seitan_b$outlier==1,]
nooutliers <- seitan_b[seitan_b$outlier==0,]

df_seitan <- df_seitan[seitan_lof < 1.3319969,]
df_seitan_outliers <- df_seitan[seitan_lof >= 1.3319969,]
```

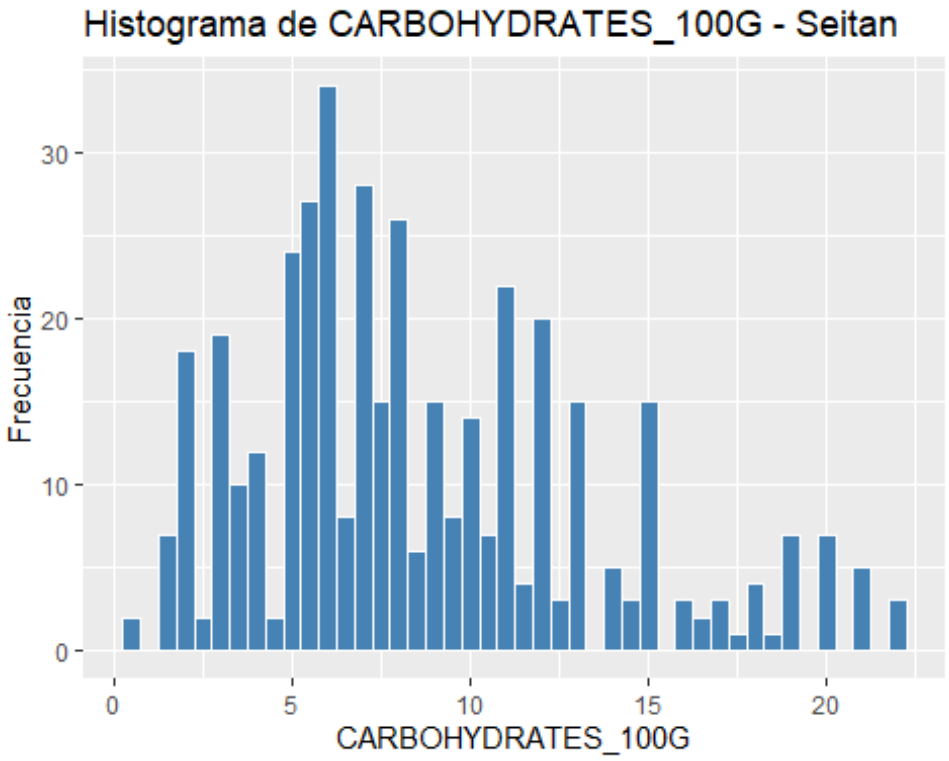
Adcionalmente, se ha inspeccionado la distribución por variable, se ha encontrando que luego de aplicar ésta técnica, todavía quedan valores extremos en las colas derechas de cada distribución, por ahora para esta entrega, se ha decidido normalizar las observaciones y luego excluir los registros con 2 desviaciones típicas superiores por variable. Los resultados se pueden apreciar tanto en el conjunto original así como normalizado en los siguientes histogramas, luego aplicar la técnica LOF y la exclusión de registros por variable.

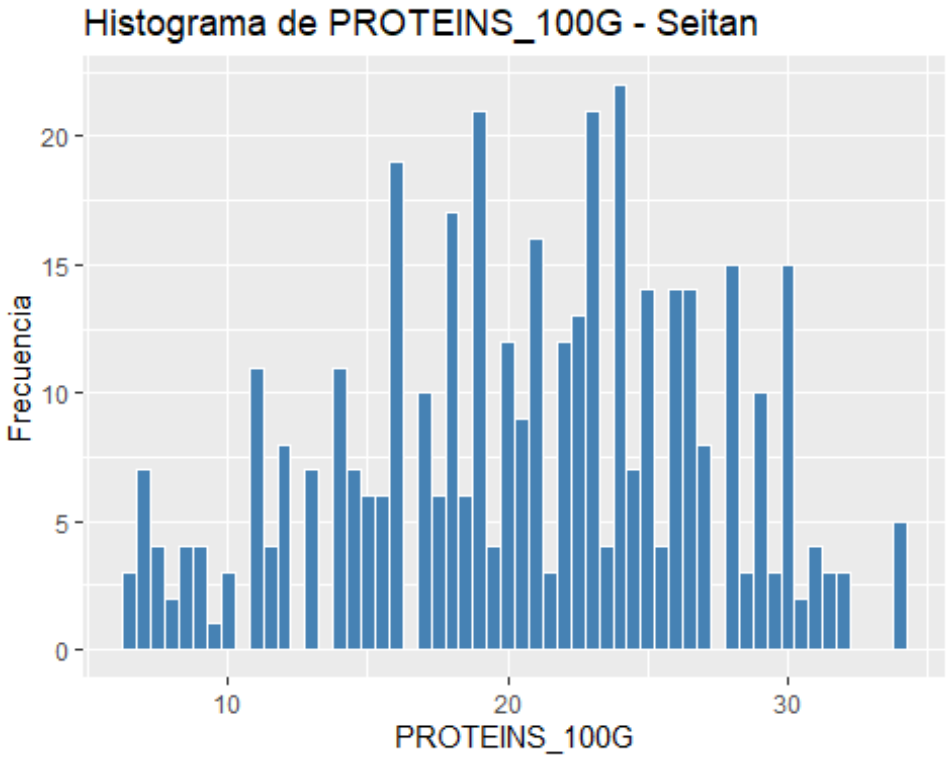
Eliminar valores atipicos univariantes

```
remove_outliers <- function(data, column, sd_threshold = 2) {
  data[abs(scale(data[[column]])) < sd_threshold, ]
}

columns_to_check <- 3:ncol(df_seitan)
for (column in columns_to_check) {
  df_seitan <- remove_outliers(df_seitan, column)
}

# Obtener las columnas cuantitativas del dataframe
columnas_cuantitativas <- sapply(df_seitan, is.numeric)

# Crear un histograma para cada columna cuantitativa
for (columna in names(df_seitan[columnas_cuantitativas])) {
  plot_data <- df_seitan[, columna]
  p <- ggplot(data.frame(x = plot_data), aes(x)) +
    geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +
    labs(title = paste("Histograma de", columna,"- Seitan"),
         x = columna,
         y = "Frecuencia")

  print(p)
}
```


Histograma de ENERGY_100G - Seitan
Frecuencia
0
3
6
9
250
500
750
1000
125
ENERGY_100G


Histograma de FAT_100G - Seitan
Frecuencia
0
10
20
30
40
50
0
5
10
15
FAT_100G





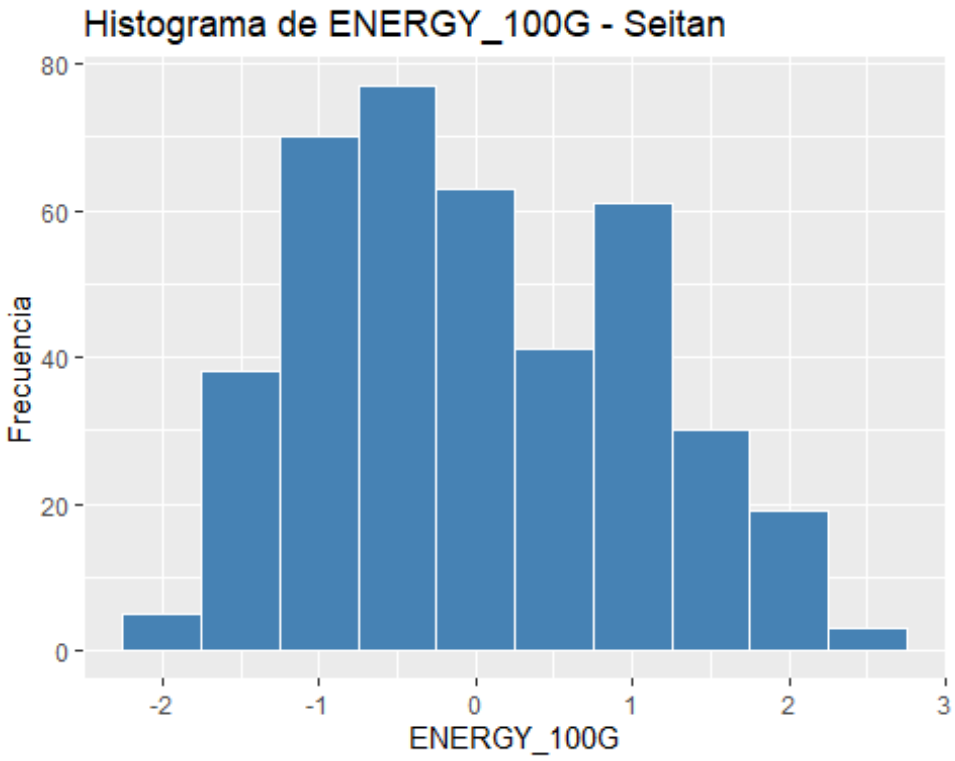

```
summary(as.data.frame(scale(df_seitan[,-1:-2])))
```

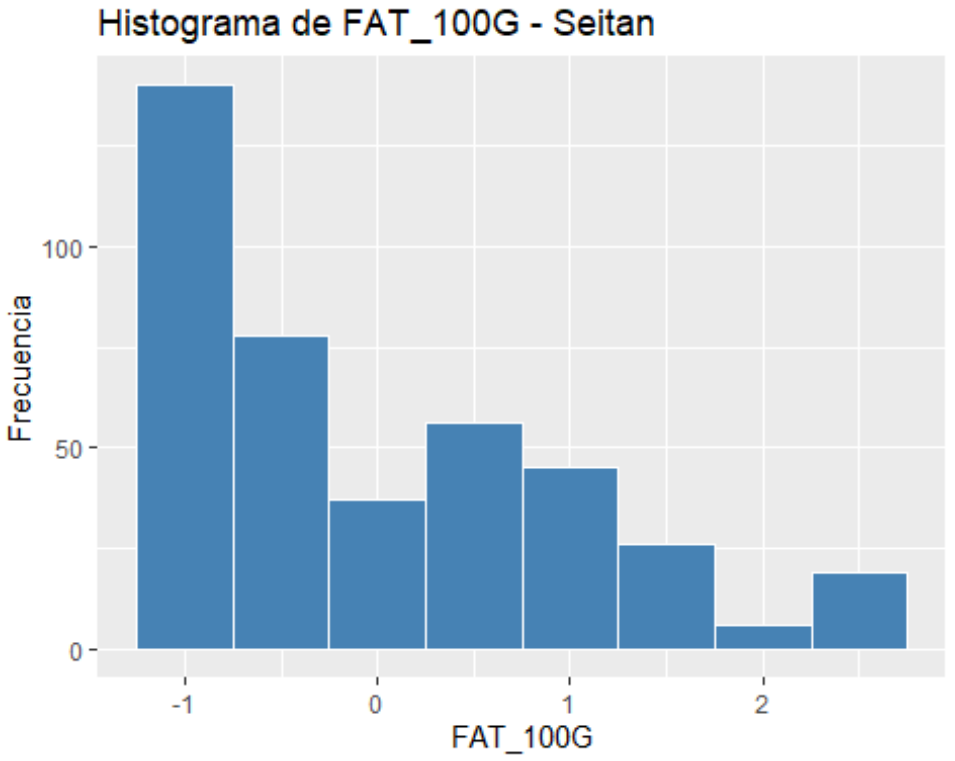
```
##   ENERGY_100G          FAT_100G        CARBOHYDRATES_100G PROTEINS_100G
##  Min.   :-2.1889   Min.   :-1.1914   Min.   :-1.7133    Min.   :-2.2511
```

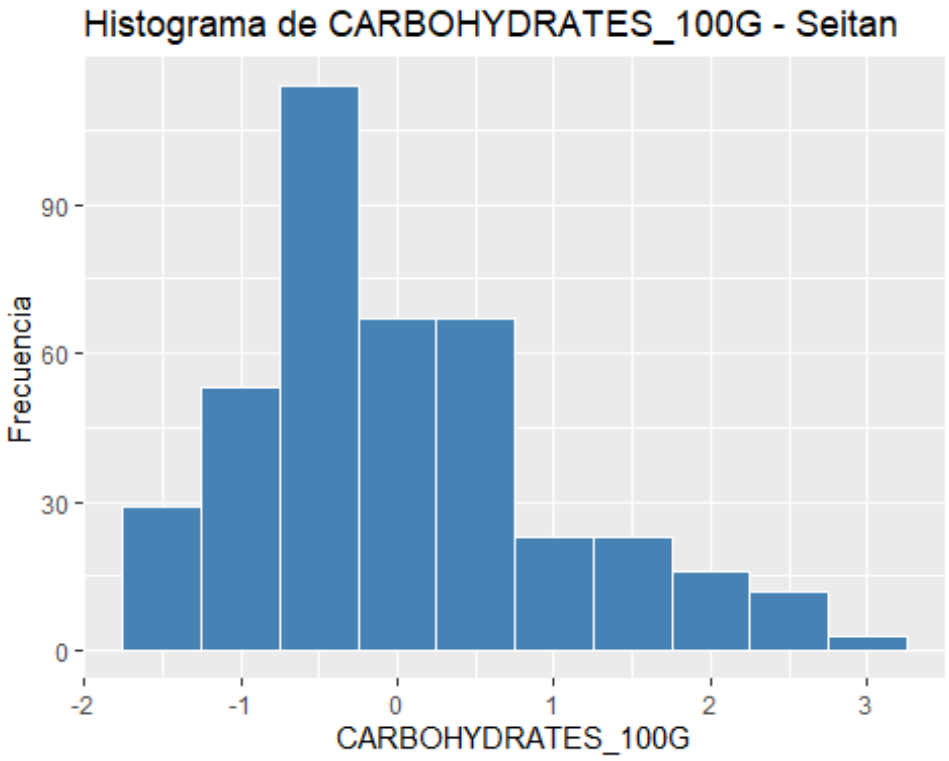
```
##  1st Qu.:-0.8173   1st Qu.:-0.8407   1st Qu.:-0.6554    1st Qu.:-0.7415
##  Median :-0.1246   Median :-0.3731   Median :-0.2111    Median : 0.0448
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.: 0.8775   3rd Qu.: 0.7257   3rd Qu.: 0.5284    3rd Qu.: 0.7446
##  Max.   : 2.3322   Max.   : 2.4557   Max.   : 2.8357    Max.   : 2.0891
```

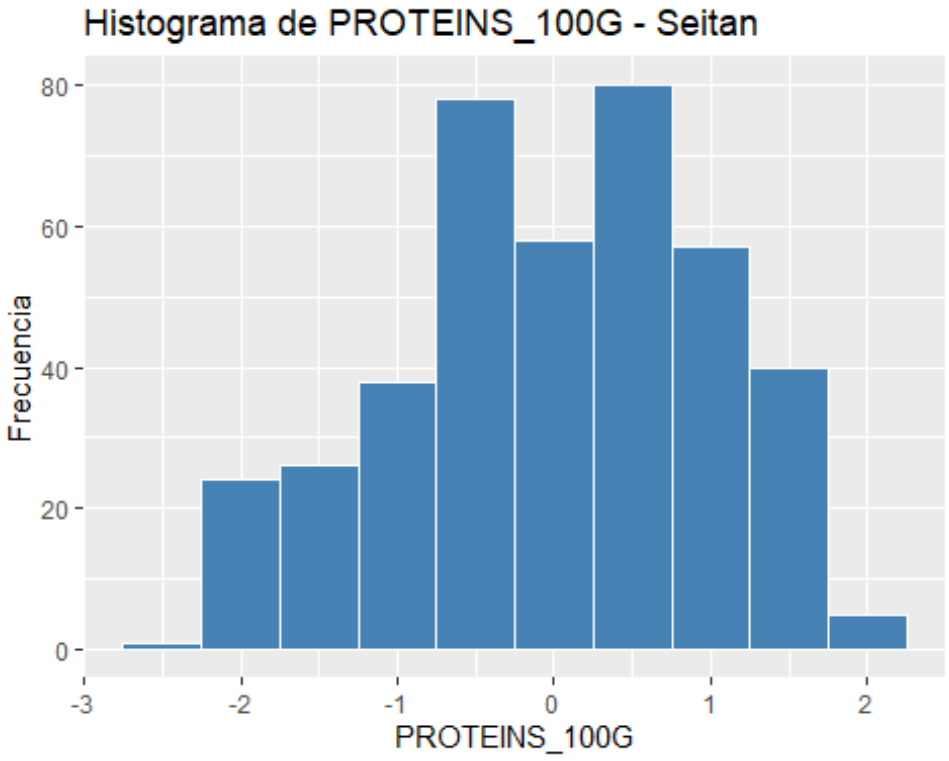
Datos normalizados para el Seitán

```
s_seitan <- as.data.frame(scale(df_seitan[,-1:-2]))
# Obtener las columnas cuantitativas del dataframe
columnas_cuantitativas <- sapply(s_seitan, is.numeric)

# Crear un histograma para cada columna cuantitativa
for (columna in names(s_seitan[columnas_cuantitativas])) {
  plot_data <- s_seitan[, columna]
  p <- ggplot(data.frame(x = plot_data), aes(x)) +
    geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +
    labs(title = paste("Histograma de", columna,"- Seitan"),
         x = columna,
         y = "Frecuencia")

  print(p)
}
```


Histograma de ENERGY_100G - Seitan
Frecuencia
80
60
40
20
0
-2
-1
0
1
2
3
ENERGY_100G

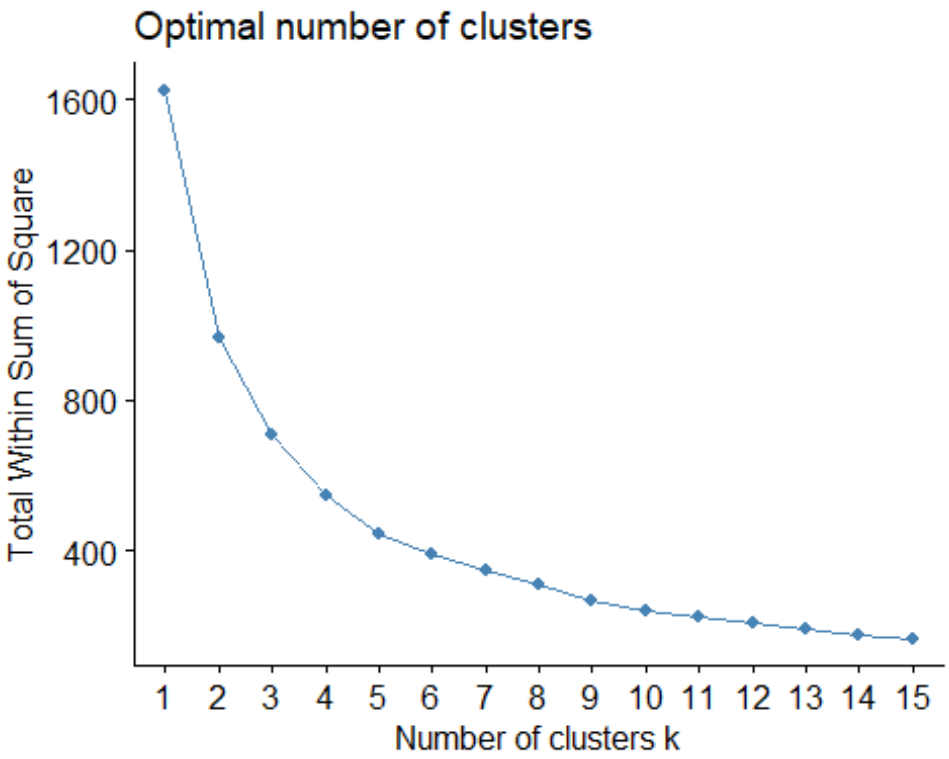

Histograma de FAT_100G - Seitan
Frecuencia
100
50
0
-1
0
1
2
FAT_100G





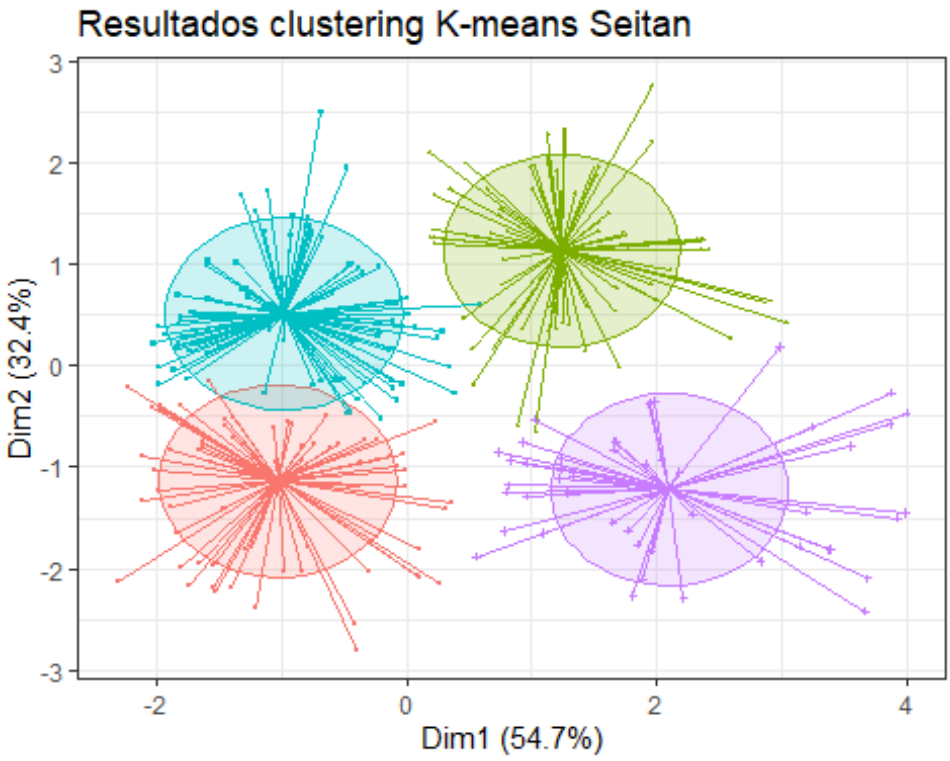
Seguidamente, se ha procedido a aplicar la técnica combinada K-means y PCA, a continuación se muestran los resultados para el gráfico del codo, el cual sugiere el total de cluster adecuado para el análisis.

```
s_seitan <- scale(df_seitan[,-1:-2])

# total de cluster óptimos
elbow <- fviz_nbclust(x = s_seitan, FUNcluster = kmeans, method = "wss",
k.max = 15,
             diss = get_dist(s_seitan, method = "euclidean"), nstart = 25)
print(elbow)
```



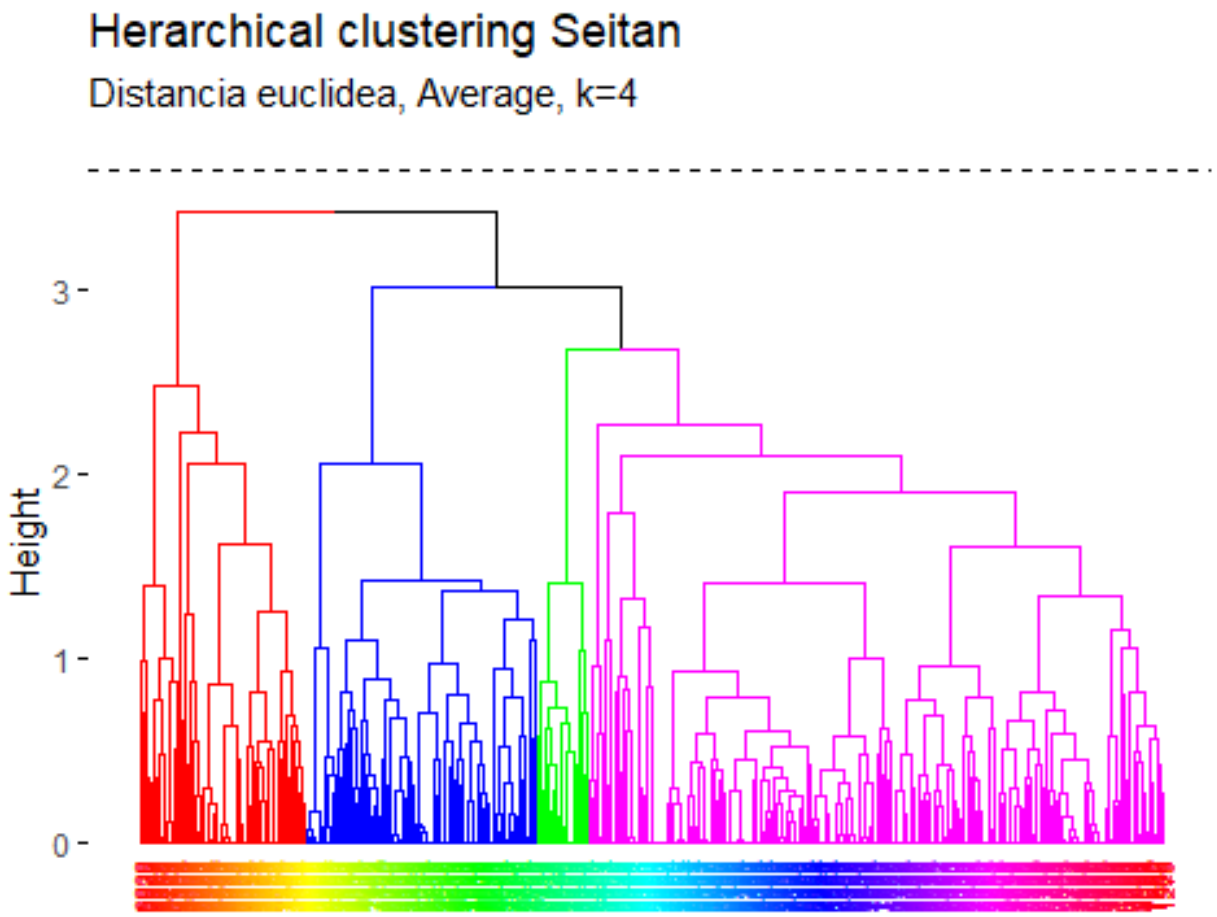
Se ha elegido 4 como potenciales cluster. Se puede apreciar como las dos primeras componentes explican el 87% de variabilidad total.

```
set.seed(123)
km_clusters <- kmeans(x = s_seitan, centers = 4, nstart = 50)

fviz_cluster(object = km_clusters, data = s_seitan, show.clust.cent = TRUE,
             ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,
             pointsize = 0.5, outlier.color = "darkred", geom = "point") +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Resultados clustering K-means Seitan")
```



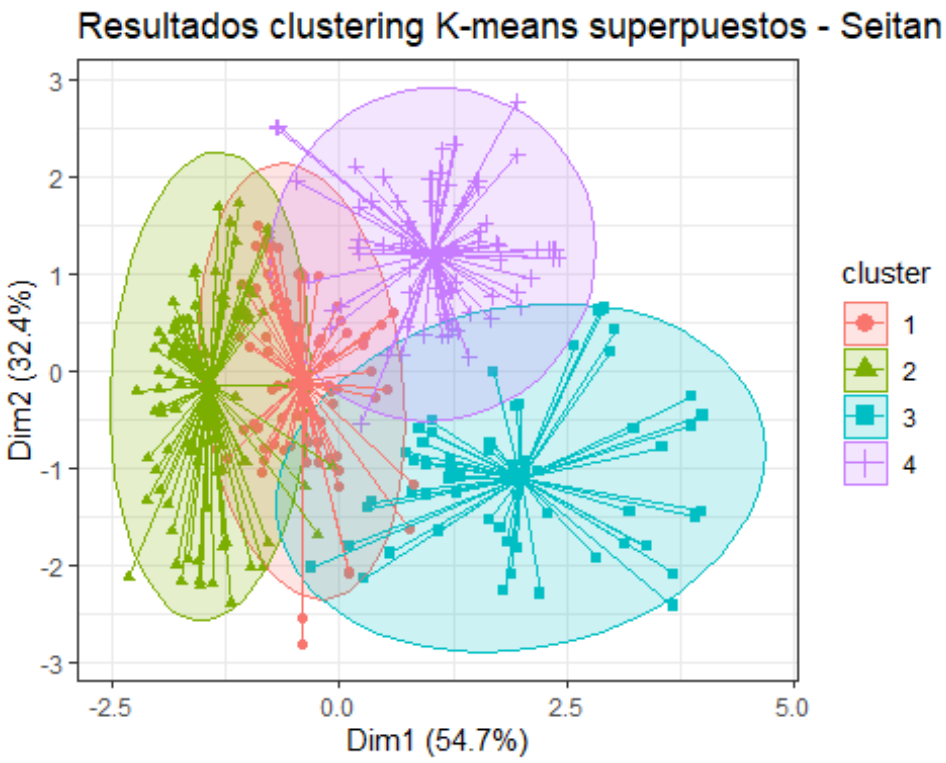
A continuación se muestra el dendrograma generado mediante el método de la distancia euclidea para 4 clusters definidos.

```
set.seed(101)

hc_euclidea_av <- hclust(d = dist(x = s_seitan, method = "euclidean"),
                         method = "average")
fviz_dend(x = hc_euclidea_av, k = 4, cex = 0.5,
          k_colors = c("red","blue","green","magenta"),color_labels_by_k = T,
          lwd = 0.2,type = "r",label_cols = rainbow(nrow(df_seitan)),
          rect_lty = "lightblue") +
  geom_hline(yintercept = 3.65, linetype = "dashed") +
  labs(title = "Herarchical clustering Seitan",
       subtitle = "Distancia euclidea, Average, k=4")
```



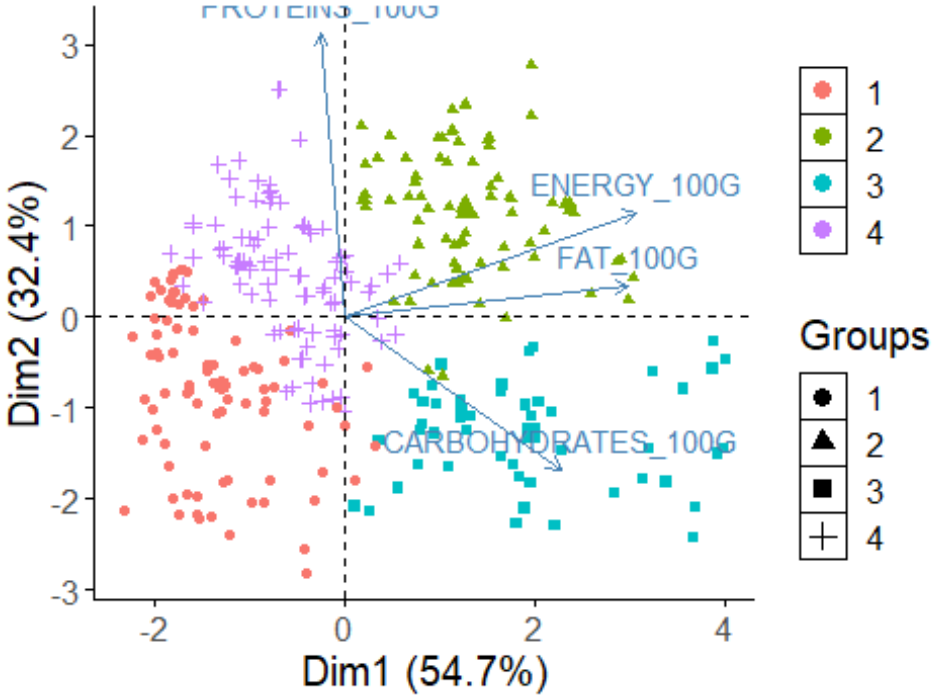
Adicionalmente, se presenta el gráficos de cluster dónde se observan los cluster que podrían estar superpuestos, esto conlleva a repetir este proceso de manera iterativa, desde la limpieza del conjunto de datos, la detección de productos atípicos, así como la definición de parámetros para identificar la mejor mamera de segmentar estos productos.

```
pam.res <- pam(s_seitan, 4)
# Visualización
fviz_cluster(pam.res, geom = "point", ellipse.type = "norm",
             show.clust.cent = TRUE,star.plot = TRUE)+
  labs(title = "Resultados clustering K-means superpuestos - Seitan")+
theme_bw()
```



Biplot PCA y K-Means para medir representatividad seitan

```
# PCA
pca <- prcomp(df_seitan[,-1:-2], scale=TRUE)
df_seitan.pca <- pca$x
# Cluster over the three first PCA dimensions
kc <- kmeans(df_seitan.pca[,1:3], 4)
fviz_pca_biplot(pca, label="var", habillage=as.factor(kc$cluster)) +
  labs(color=NULL) + ggtitle("") +
  theme(text = element_text(size = 15),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        legend.key = element_rect(fill = "white"))
```



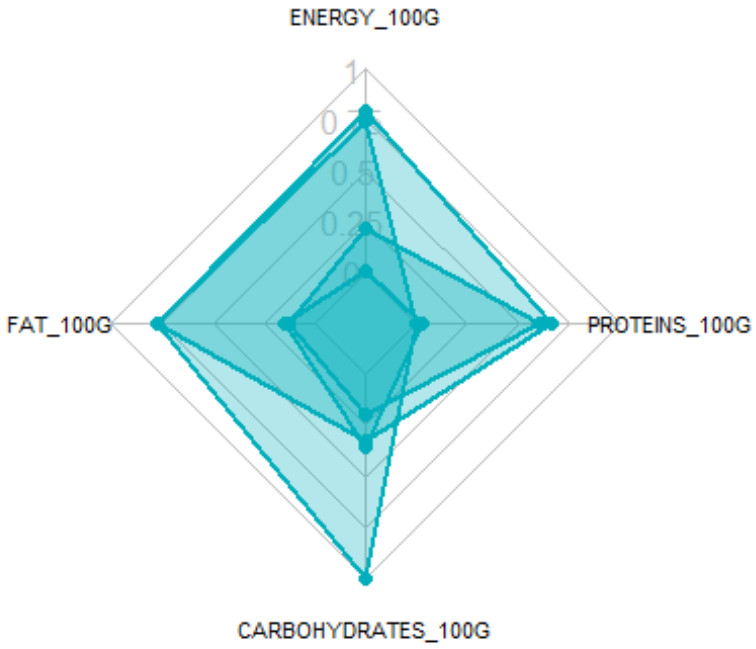
El gráfico previo ilustra los productos en el plano, se observa primero como las variables ENERGY y FAT se encuentran altamente correlacionadas con el eje 1, por otra parte, PROTEINS esta correlacionada con el eje 2, finalmente CARBOHYDRATES esta explicada parcialmente por el eje 1 y 2. Productos cercanos a la dirección de los vectores indican valores altos en esas variables, productos opuestos a la dirección de los vectores indican bajos valores en esas variables. En este sentido, se puede identificar lo siguiente:

- Grupo 1. Productos de seitán con valores bajos en energía, grasas y proteínas pero con ligero grado de carbohidratos.
- Grupo 2. Productos con valores altos en grasa, proteínas y energía pero bajos en carbohidratos.
- Grupo 3. Productos con valores altos en carbohidratos grasas y energías pero con valores bajos en proteínas.
- Grupo 4. Productos de seitán con valore altos en proteínas, valore bajos en grasa y energía y ligeramente bajos en carbohidratos.

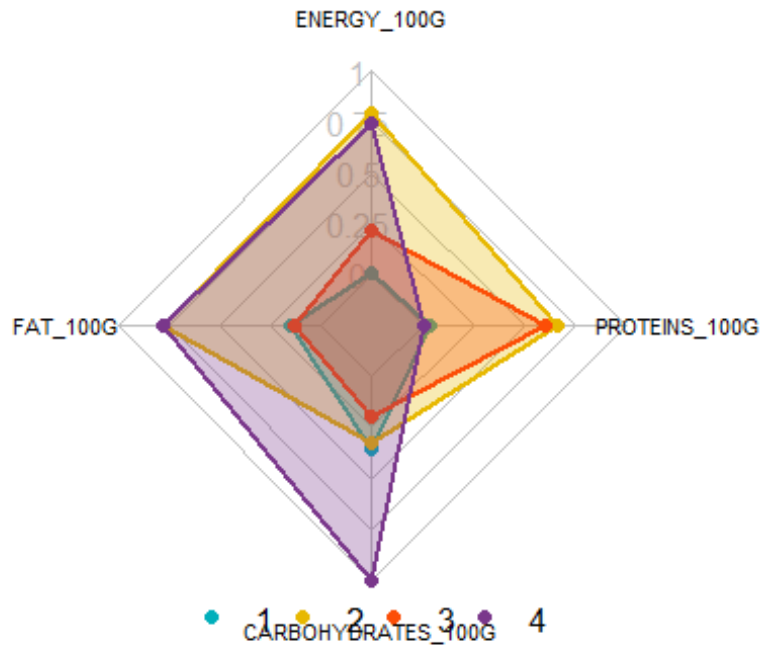
A continuación se muestran estos resultados de otra manera mediante gráficos de radar:

```
library(ggplot2)

# Obtener los centroides de cada clúster
centroids <- as.data.frame(km_clusters$centers)


# Obtener el valor mínimo y máximo en el dataframe
min_value <- min(centroids, na.rm = TRUE)
```

```r
max_value <- max(centroids, na.rm = TRUE)

# Escalar los datos entre 0 y 1
scaled_df <- (centroids - min_value) / (max_value - min_value)

library(fmsb)

## Warning: package 'fmsb' was built under R version 4.3.1

# Define the variable ranges: maximum and minimum
max_min <- data.frame(
  ENERGY_100G = c(1, 0), FAT_100G = c(1, 0), CARBOHYDRATES_100G = c(1,
0),PROTEINS_100G = c(1, 0)
)
rownames(max_min) <- c("Max", "Min")

# Bind the variable ranges to the data
df <- rbind(max_min, scaled_df)

create_beautiful_radarchart <- function(data, color = "#00AFBB",
                                        vlabels = colnames(data), vlcex =
0.7,
                                        caxislabels = NULL, title = NULL,
...){
  radarchart(
    data, axistype = 1,
    # Customize the polygon
    pcol = color, pfcol = scales::alpha(color, 0.3), plwd = 2, plty = 1,
    # Customize the grid
    cglcol = "grey", cglty = 1, cglwd = 0.8,
    # Customize the axis
    axislabcol = "grey",
    # Variable labels
    vlcex = vlcex, vlabels = vlabels,
    caxislabels = caxislabels, title = title, ...
  )
}

# Reduce plot margin using par()
op <- par(mar = c(1, 2, 2, 1))
create_beautiful_radarchart(df, caxislabels = c(0, .25, .5, .75, 1))
```




```
par(op)

# Reduce plot margin using par()
op <- par(mar = c(1, 2, 2, 2))
# Create the radar charts
create_beautiful_radarchart(
  data = df, caxislabels = c(0, .25, .5, .75, 1),
  color = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B")
)
# Add an horizontal legend
legend(
  x = "bottom", legend = rownames(df[-c(1,2),]), horiz = TRUE,
  bty = "n", pch = 20 , col = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B"),
  text.col = "black", cex = 1, pt.cex = 1.5
  )
```

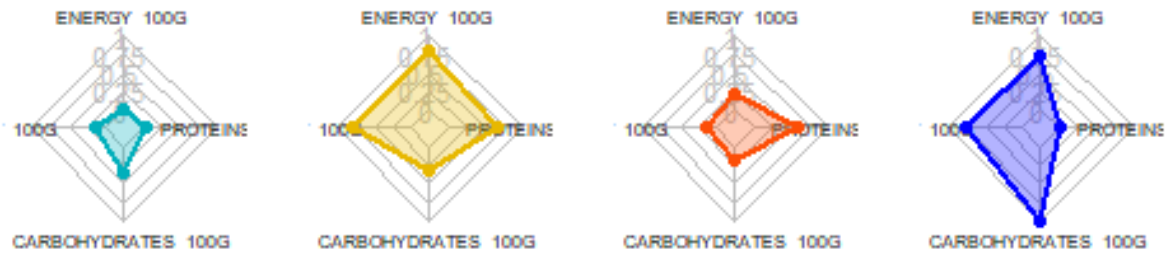


- Cluster 1. Productos de seitán con valores bajos en todas las variables pero con ligero grado de carbohidratos.
- Cluster2. Productos con valores altos en grasa, proteínas y energía pero bajos en carbohidratos.
- Cluster 3. Productos de seitán con valore altos en proteínas, valore bajos en grasa y energía y ligeramente bajos en carbohidratos.
- Cluster4. Productos con valores altos en carbohidratos grasas y energías pero con valores bajos en proteínas.

Estos resultados se corresponden con los apreciados en el Biplot PCA y K-Means.

```
par(op)

# Define colors and titles
colors <- c("#00AFBB", "#E7B800", "#FC4E07","blue")
titles <- c("1", "2", "3","4")

# Reduce plot margin using par()
# Split the screen in 3 parts
op <- par(mar = c(1, 1, 1, 1))

par(mfrow = c(1,4))

# Create the radar chart
for(i in 1:4){
  create_beautiful_radarchart(
```

```
    data = df[c(1, 2, i+2), ], caxislabels = c(0, .25, .5, .75, 1),
    color = colors[i], title = titles[i]
    )
}
```




```
par(op)
```

A continación se presenta un listado de los productos mas cercanos de cada centroide.

Visualizar los productos más cercanos al centroide que representan cada cluster top10 para el Seitán

```
# Realizar clustering en el dataframe
set.seed(123)
df_top <- df_seitan
km_clusters <- kmeans(x = df_top[, c("ENERGY_100G", "FAT_100G",
"CARBOHYDRATES_100G", "PROTEINS_100G")], centers = 4, nstart = 50)

# Obtener las asignaciones de clúster
cluster_assignments <- km_clusters$cluster

# Agregar las asignaciones de clúster al dataframe
df_top$cluster <- cluster_assignments

# Inicializar una lista para almacenar los productos más cercanos a cada
centroide
closest_products <- vector("list", max(cluster_assignments))

# Encontrar los productos más cercanos a cada centroide
for (cluster in 1:max(cluster_assignments)) {
  cluster_center <- km_clusters$centers[cluster, ]
  distances <- apply(df_top[, c("ENERGY_100G", "FAT_100G",
"CARBOHYDRATES_100G", "PROTEINS_100G")], 1, function(row) {
    sum((row - cluster_center)^2)
  })
  closest_products[[cluster]] <- head(order(distances), 10)
}
```

```
# Imprimir los productos más cercanos a cada centroide
for (cluster in 1:max(cluster_assignments)) {
  cat("Cluster", cluster, ":\n")
  print(df_top[closest_products[[cluster]], c("PRODUCT_NAME", "ENERGY_100G",
"FAT_100G", "CARBOHYDRATES_100G", "PROTEINS_100G")])
  cat("\n")
}
```

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
7673	CROCCHETTA DI SEITAN	649	2.0	14.0	19
2116	ARROSTO DI SEITAN	653	1.4	12.0	23
5972	SEITAN VEGGIE-HACHE	653	1.5	13.0	21
5977	SEITAN MET MARROKAANSE KRUIDEN	653	1.5	13.0	21
8523	BIO BURGUER VEGETAL SEITAN AL CURRY	649	6.9	9.8	11
8538	BIO MAXI BURGUER VEGETAL, SEITAN AL CURRY	649	6.9	9.8	11
5708	SEITANGEHAKT	636	0.9	7.8	28
1538	SEITAN	636	1.0	5.6	30
4363	SEITAN, DURCHSCHNITT	636	1.0	5.6	30
6031	VEGAN BURGER SEITAN WITH CHEESE FLAVOR	657	5.0	13.0	14

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
8547	SALCHIVEGGIE SEITAN Y TOFU	1071	12.0	10.0	26.0
4784	BISTECCA DI SEITAN BIO ALLA MEDITERRANEA	1059	14.0	11.0	19.0
7869	AFFETTATO VEGETALE DI SEITAN E TOFU	1075	14.7	7.5	23.0
7864	AFFETTATO VEGETALE DI SEITAN E TOFU	1075	15.0	7.6	23.0
1976	SEITAN TOFU	1079	14.8	6.8	23.1
8535	BIO SEITAN Y TOFU TRADICIONAL	1084	15.0	6.8	23.0
8385	SEITAN CON TOFU	1086	15.0	6.9	23.0
8536	SEITAN Y TOFU TRADICIONAL	1088	15.0	6.8	23.0
8795	SEITAN BIO	1088	15.0	6.8	23.0
7651	CHARCUTERIE DE SEITAN	1042	12.0	9.9	24.0

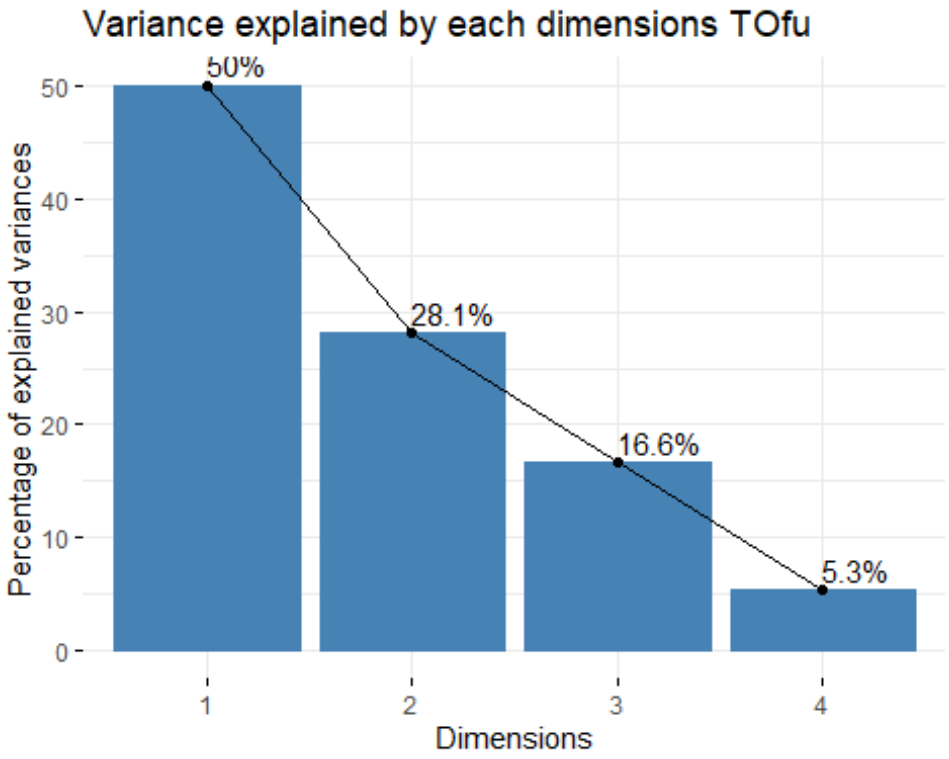
	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
8787	SEITAN NATURAL VEGANO	444	1.00	4.200000	20.00000
792	CUBED SEITAN WHEAT PROTEIN	443	0.00	4.705882	21.17647
7870	SEITAN INCREDIBILE	448	1.10	8.300000	16.00000
7515	SEITAN BIOLOGICO AL NATURALE	448	1.10	8.300000	16.00000
453	SEITAN	448	0.77	9.520000	17.86000
6033	SEITAN NATURAL	452	1.30	1.900000	21.50000
2239	SEITAN GOURMET	456	1.50	6.000000	17.50000
2235	SEITAN GOUTMET	456	1.50	6.000000	18.00000
2236	SEITAN GOURMET ORIGINAL	456	1.50	6.000000	18.00000
2241	SEITAN GOURMET	456	1.50	6.000000	18.00000

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
2164	SEITAN ALLA PIASTRA	879	7.7000	11.7000	23.3000
1493	SEITAN	865	8.6667	10.0000	22.0000
4913	SEITAN AUFSCHNITT SPICY BEANS	864	6.8000	9.3000	25.2000
2183	COTOLETTA DI SEITAN DORATA	879	4.0000	17.0000	18.0000
4458	SEITAN	883	9.4000	8.6000	22.4000
4910	SEITAN GESCHNETZELTES	883	6.8000	8.0000	28.0000
4921	SEITAN HACK	883	6.8000	8.0000	28.0000
4456	MEDAILLONS DE SEITAN	866	6.6000	4.1000	31.4000
1494	SEITAN VEGE-POULET	886	8.2353	10.5882	23.5294
4908	SEITAN GERAUCHERT	862	7.1000	5.0000	29.3000

02. Tofu

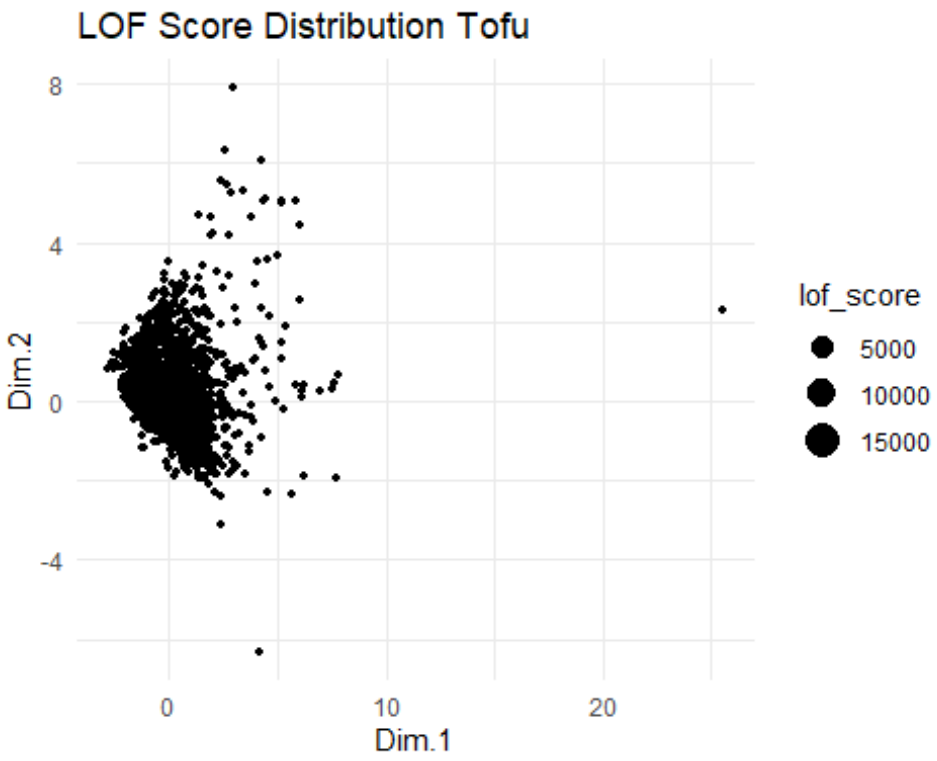
De manera análoga al seitán se ha realizado con el Tofu, las interpretaciones aplicarían de manera similar.

```
fviz_eig(tofu_pca, ncp = 6, addlabels = T, main = "Variance explained by each
dimensions TOfu")
```

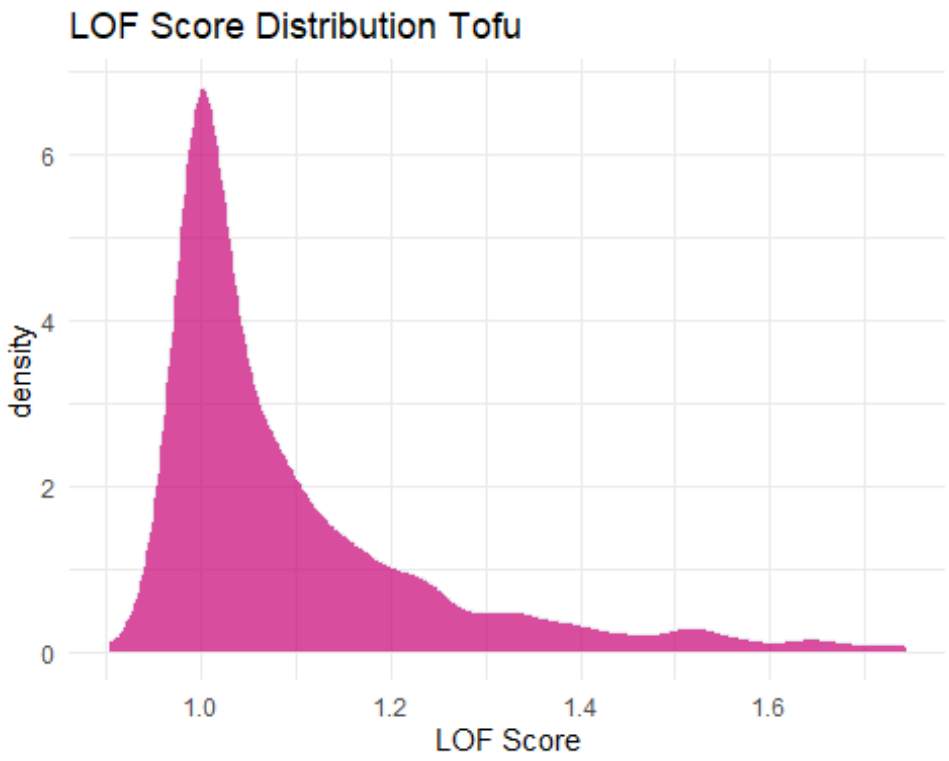


Técnica LOF - Tofu

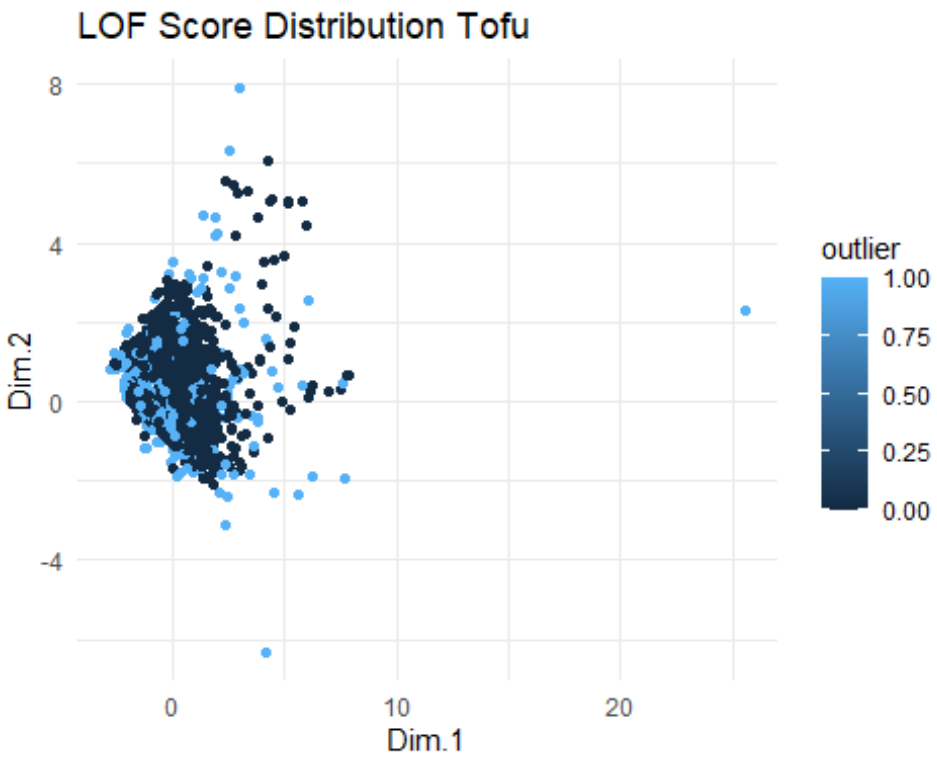
```
library(ggthemes)
tofu_a <- data.frame(tofu_pca$ind$coord[,1:3])
tofu_b <- cbind(tofu_a, lof_score = tofu_lof)
#tofu_b <- cbind(tofu_a, fraud = tofu_clean$, lof_score = tofu_clean$lof)

tofu_lof_visual <- ggplot(tofu_b, aes(x=Dim.1 ,y=Dim.2)) +
    geom_point(aes(size=lof_score)) +
  ggtitle("LOF Score Distribution Tofu")+
    theme_minimal()

tofu_lof_visual
```




```
summary(tofu_b)

##      Dim.1              Dim.2              Dim.3             lof_score
##  Min.   :-2.8170   Min.   :-6.3239   Min.   :-6.57541   Min.   :0.9036
##  1st Qu.:-0.8214   1st Qu.:-0.6949   1st Qu.:-0.25797   1st Qu.:1.0000
##  Median :-0.1787   Median :-0.2655   Median : 0.01589   Median :1.0532
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   :   Inf
##  3rd Qu.: 0.6344   3rd Qu.: 0.4581   3rd Qu.: 0.30434   3rd Qu.:1.2100
##  Max.   :25.5570   Max.   : 7.8974   Max.   : 6.35320   Max.   :   Inf

tofu_b %>%
  filter(lof_score <= 1.75) %>%
  ggplot( aes(x=lof_score)) +
    geom_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +
    scale_fill_manual(values="#8fce00") +
    xlab("LOF Score")+
  ggtitle("LOF Score Distribution Tofu")+
    theme_minimal() +
    labs(fill="")
```




```
quantile(tofu_b$lof_score, probs = c(0, 0.8))

##        0%       80%
## 0.9036087 1.2815631

tofu_b <- tofu_b %>%
  mutate(outlier = ifelse(lof_score > 1.2815631, 1, 0))

tofu_lof_visual_b <- ggplot(tofu_b, aes(x=Dim.1 ,y=Dim.2, color=outlier)) +
    geom_point() +
  ggtitle("LOF Score Distribution Tofu")+
    theme_minimal()

tofu_lof_visual_b
```

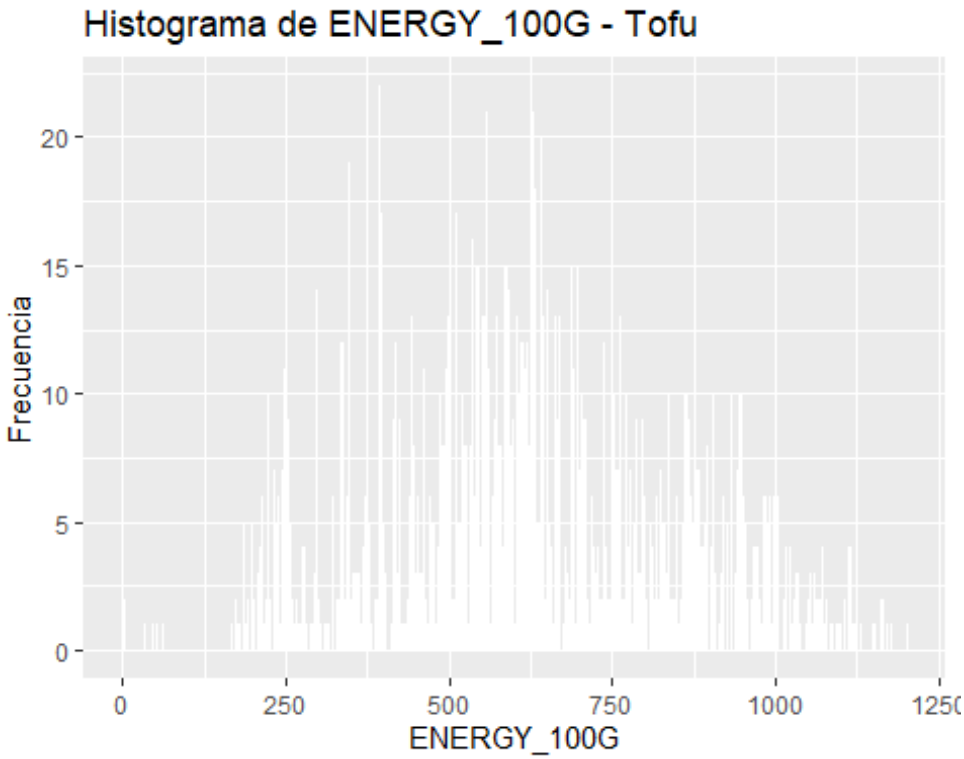


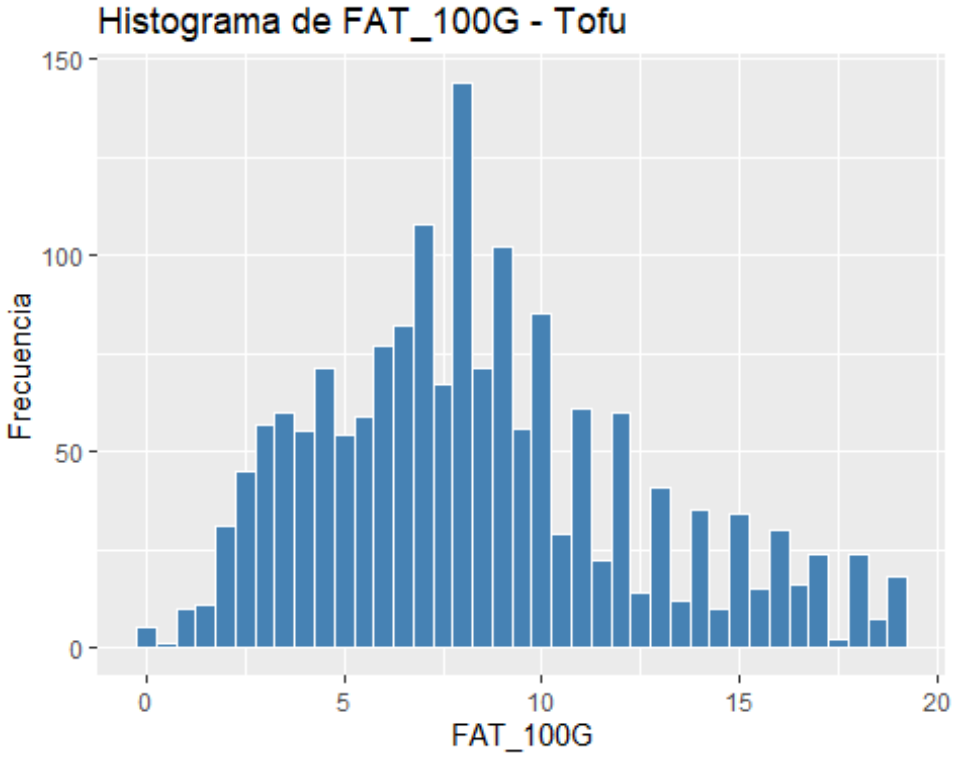

```
outliers <- tofu_b[tofu_b$outlier==1,]
nooutliers <- tofu_b[tofu_b$outlier==0,]

df_tofu <- df_tofu[tofu_lof < 1.2822731,]
df_tofu_outliers <- df_tofu[tofu_lof >= 1.2822731,]
```

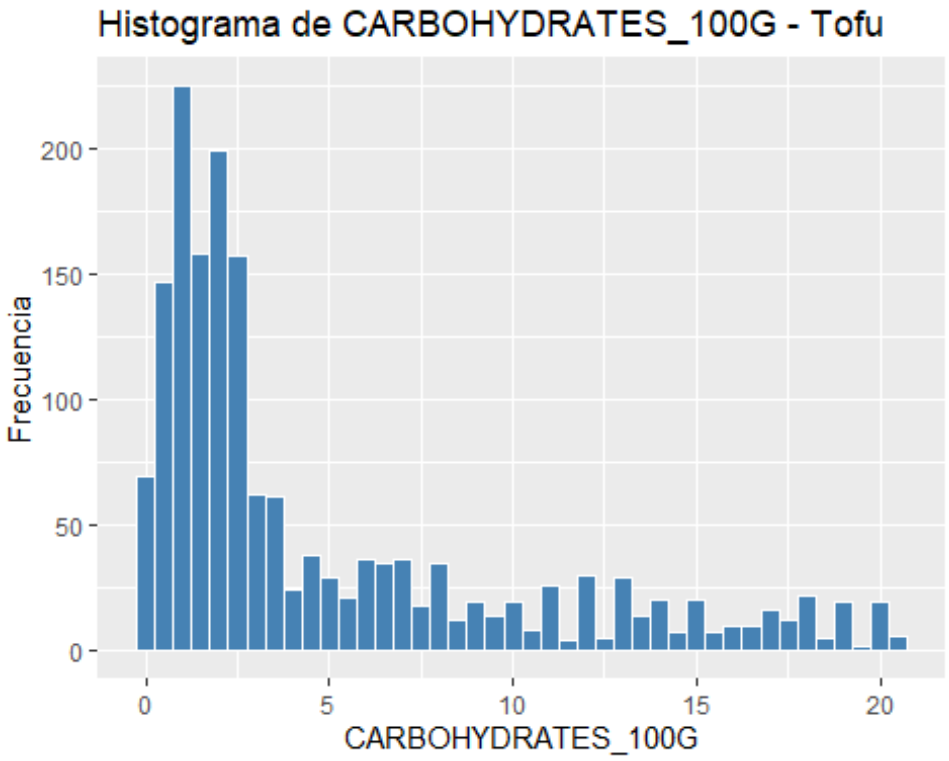
Eliminar valores atipicos univariantes

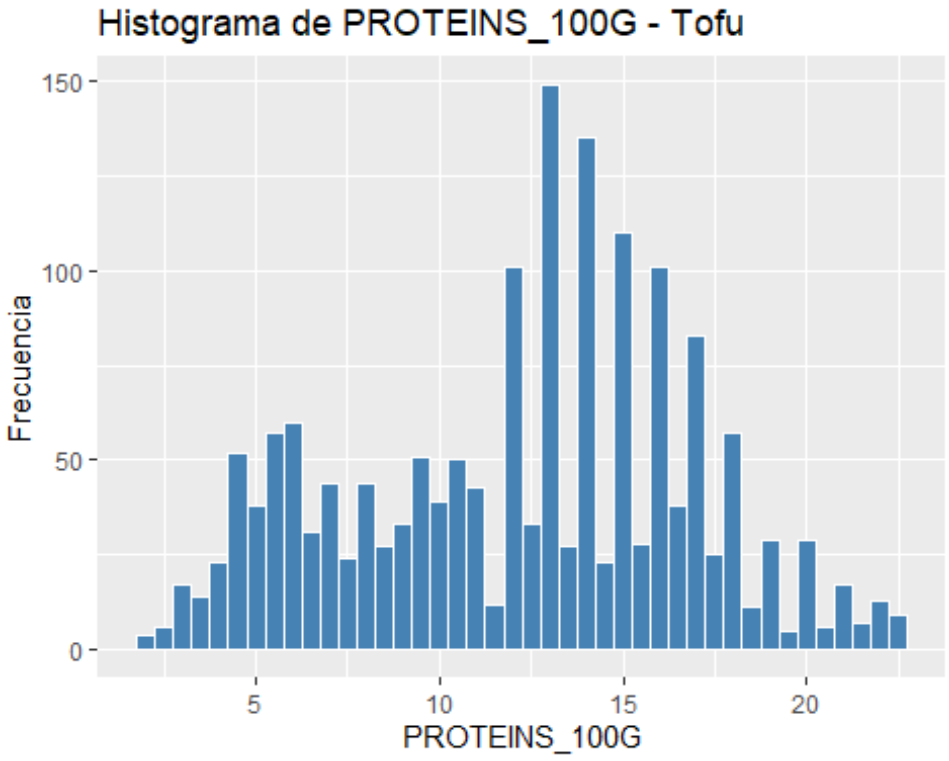
```
remove_outliers <- function(data, column, sd_threshold = 2) {
  data[abs(scale(data[[column]])) < sd_threshold, ]
}

columns_to_check <- 3:ncol(df_tofu)
for (column in columns_to_check) {
  df_tofu <- remove_outliers(df_tofu, column)
}

# Obtener las columnas cuantitativas del dataframe
columnas_cuantitativas <- sapply(df_tofu, is.numeric)

# Crear un histograma para cada columna cuantitativa
for (columna in names(df_tofu[columnas_cuantitativas])) {
  plot_data <- df_tofu[, columna]
  p <- ggplot(data.frame(x = plot_data), aes(x)) +
    geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +
    labs(title = paste("Histograma de", columna,"- Tofu"),
         x = columna,
         y = "Frecuencia")
```

```
  print(p)
}
```


Histograma de ENERGY_100G - Tofu
Frecuencia
20
15
10
5
0
0
250
500
750
1000
ENERGY_100G


Histograma de FAT_100G - Tofu
Frecuencia
150
100
50
0
0
5
10
15
20
FAT_100G





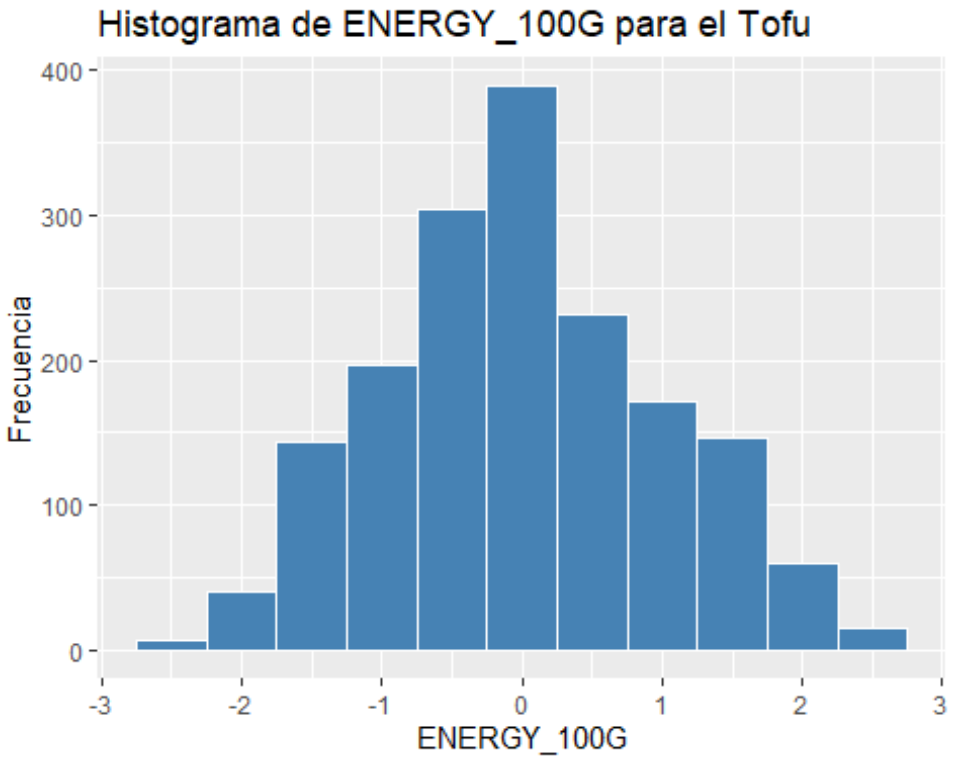

```
summary(as.data.frame(scale(df_tofu[,-1:-2])))
```

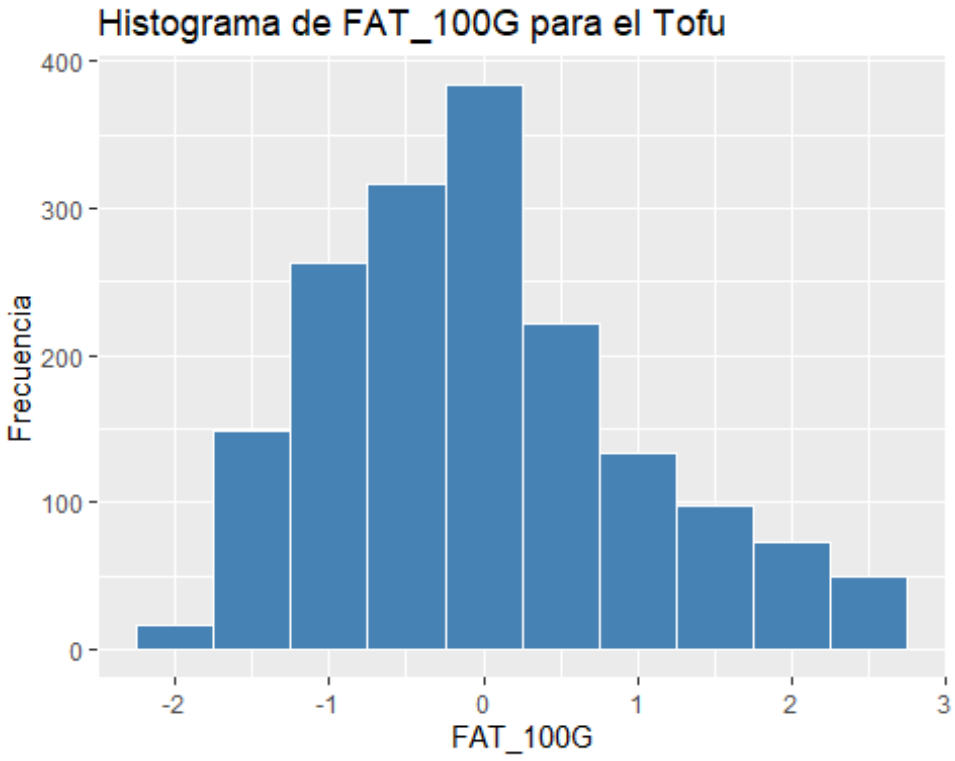
```
##    ENERGY_100G          FAT_100G        CARBOHYDRATES_100G PROTEINS_100G
##  Min.   :-2.74021   Min.   :-2.05837   Min.   :-0.9301    Min.   :-2.1914
```

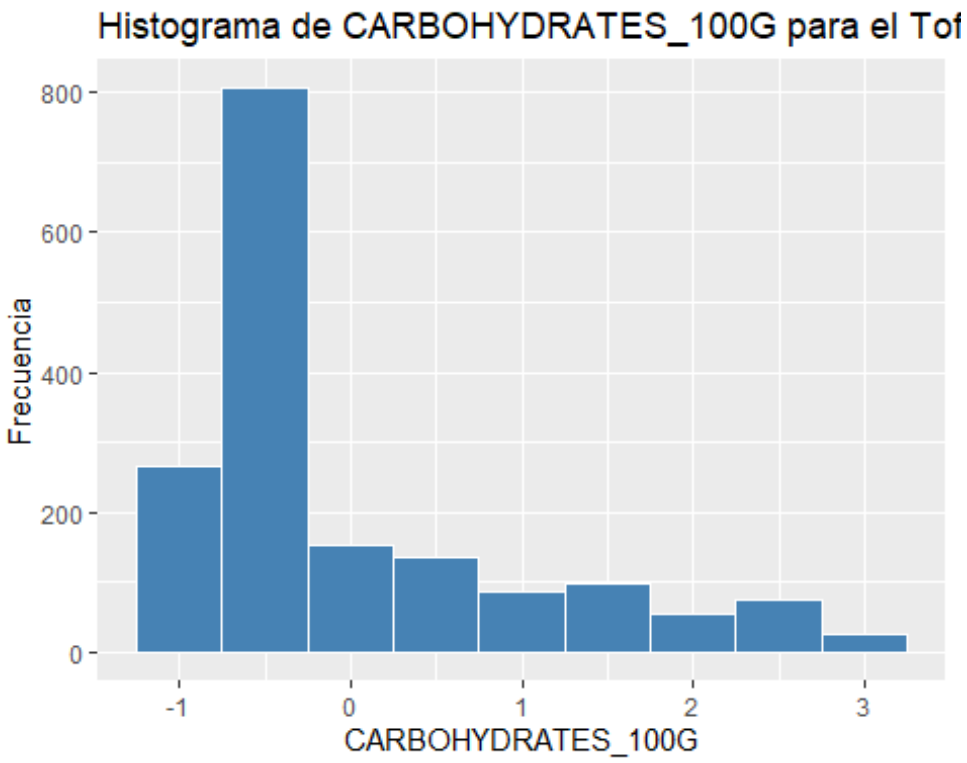
```
##  1st Qu.:-0.67723   1st Qu.:-0.75421   1st Qu.:-0.7025    1st Qu.:-0.8305
##  Median :-0.05834   Median :-0.09722   Median :-0.4839    Median : 0.1723
##  Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.: 0.67267   3rd Qu.: 0.56711   3rd Qu.: 0.3973    3rd Qu.: 0.7740
##  Max.   : 2.64594   Max.   : 2.59936   Max.   : 2.9574    Max.   : 2.2136
```

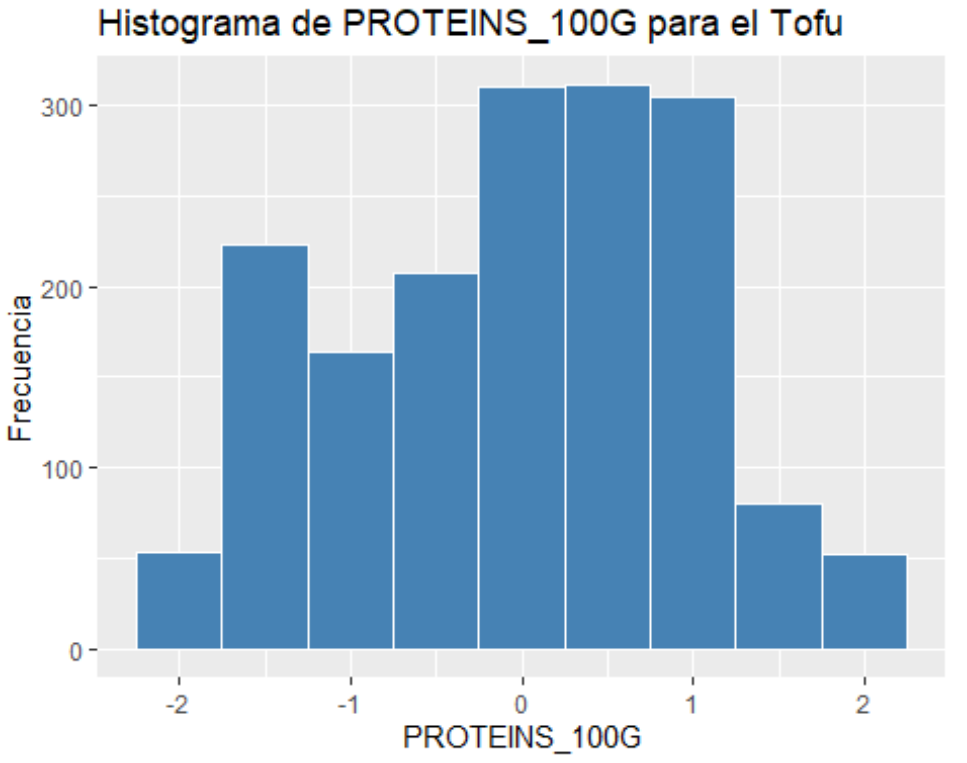
Datos normalizados para el tofu

```
s_tofu <- as.data.frame(scale(df_tofu[,-1:-2]))
# Obtener las columnas cuantitativas del dataframe
columnas_cuantitativas <- sapply(s_tofu, is.numeric)

# Crear un histograma para cada columna cuantitativa
for (columna in names(s_tofu[columnas_cuantitativas])) {
  plot_data <- s_tofu[, columna]
  p <- ggplot(data.frame(x = plot_data), aes(x)) +
    geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +
    labs(title = paste("Histograma de", columna,"para el Tofu"),
         x = columna,
         y = "Frecuencia")

  print(p)
}
```


Histograma de ENERGY_100G para el Tofu
Frecuencia
400
300
200
100
0
-3
-2
-1
0
1
2
3
ENERGY_100G

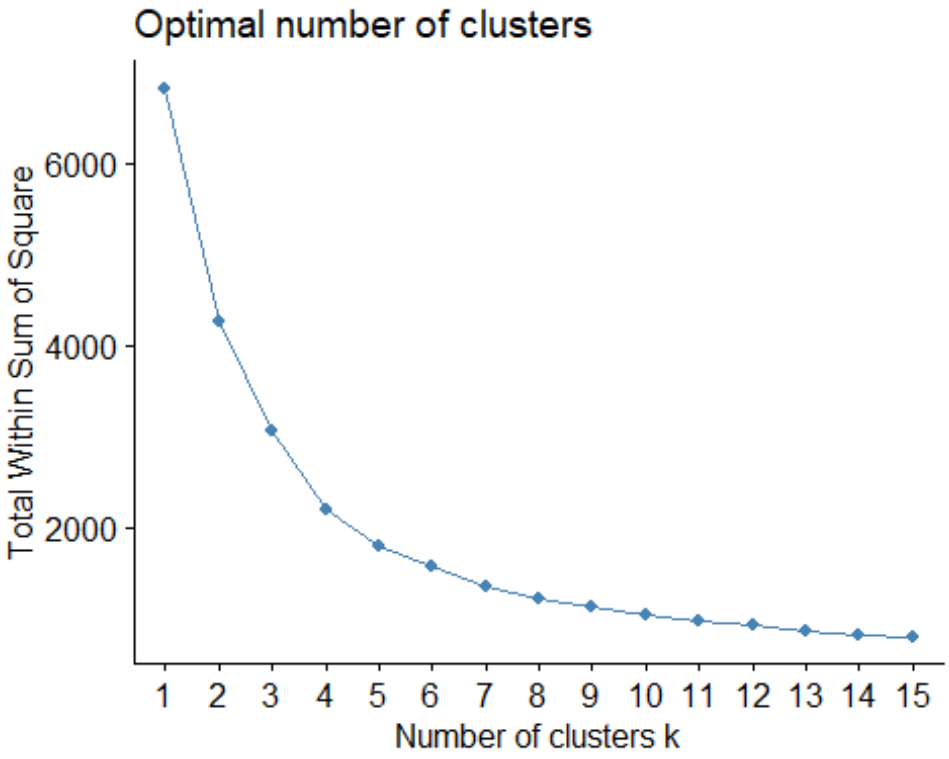

Histograma de FAT_100G para el Tofu
Frecuencia
400
300
200
100
0
-2
-1
0
1
2
3
FAT_100G



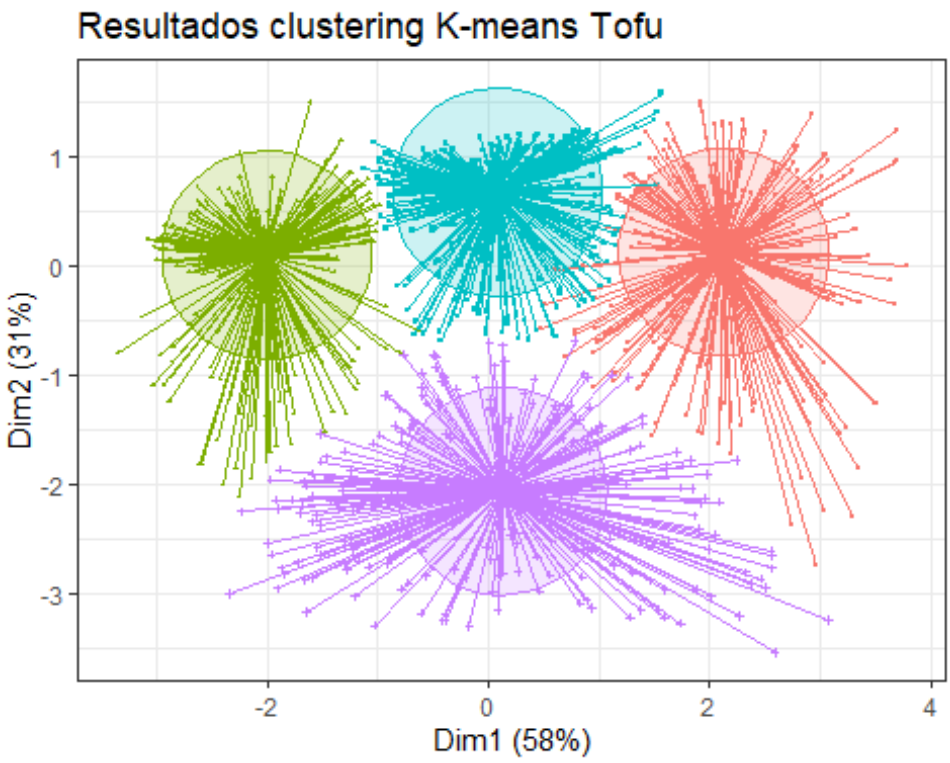



```
s_tofu <- scale(df_tofu[,-1:-2])

# total de cluster óptimos
elbow <- fviz_nbclust(x = s_tofu, FUNcluster = kmeans, method = "wss", k.max
```

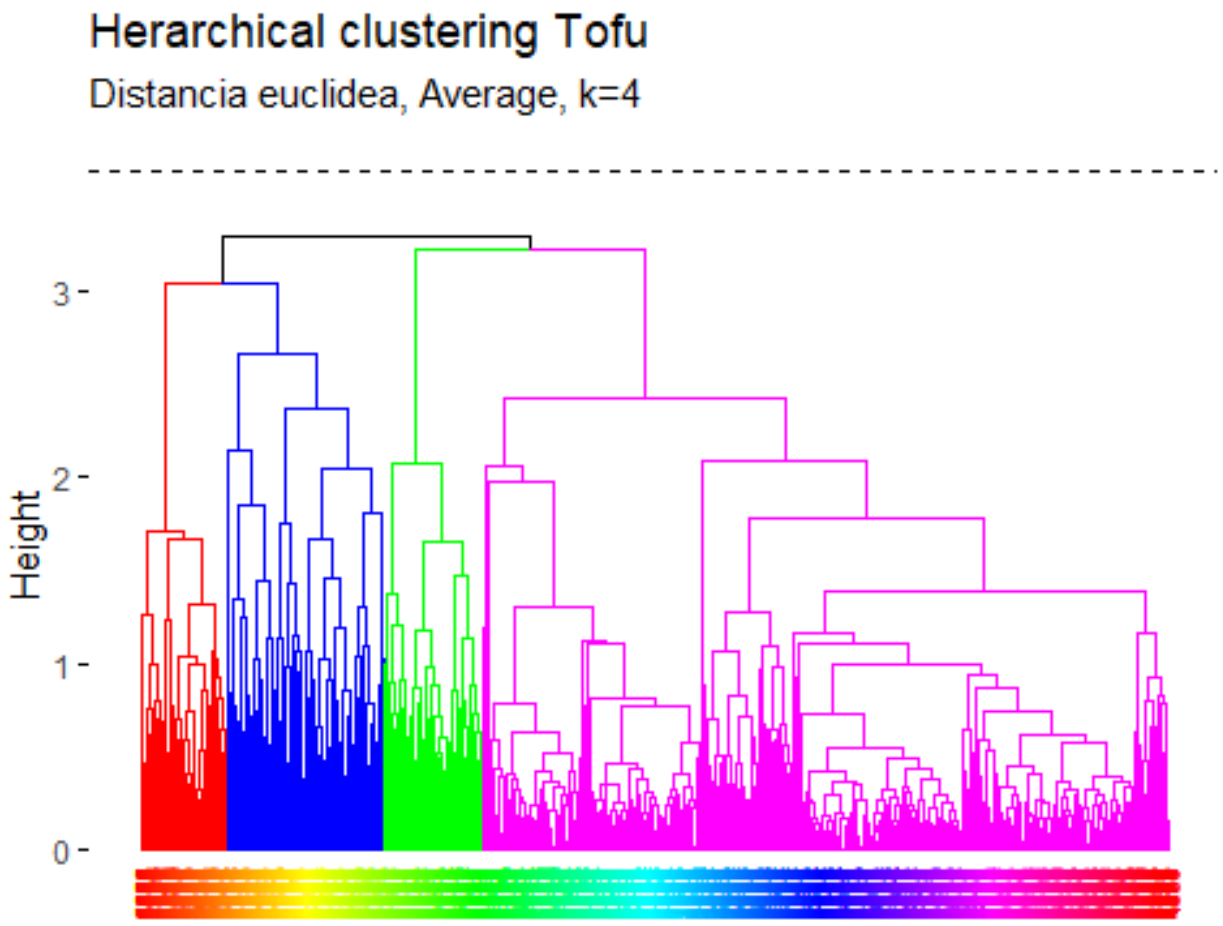
```
= 15,
              diss = get_dist(s_tofu, method = "euclidean"), nstart = 25)
print(elbow)
```



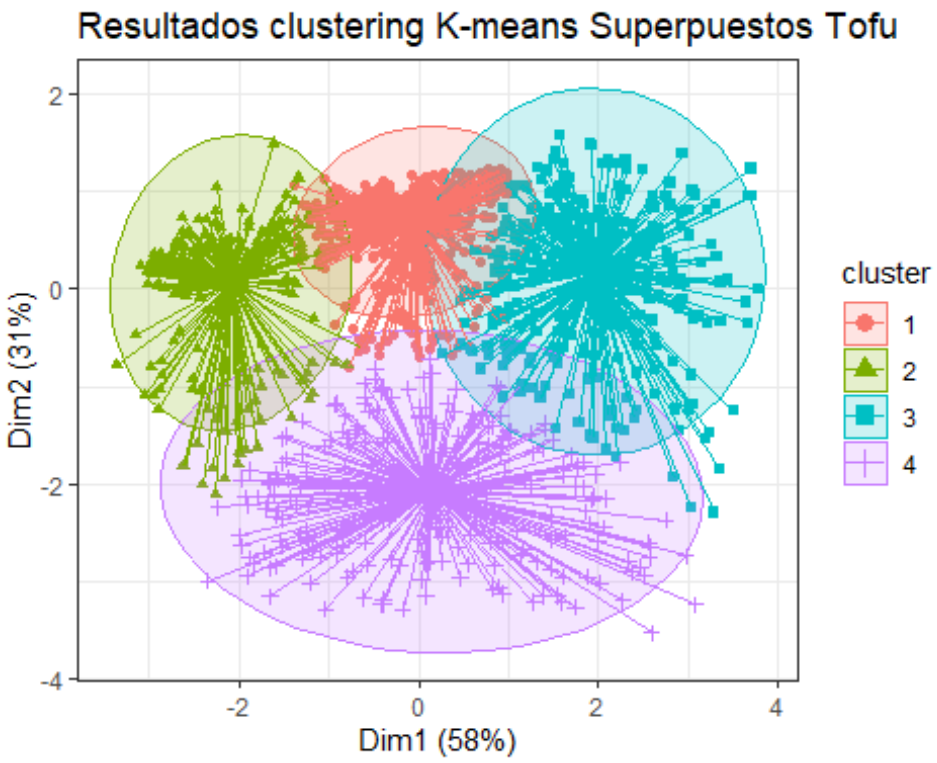

```
set.seed(123)
km_clusters <- kmeans(x = s_tofu, centers = 4, nstart = 25)

fviz_cluster(object = km_clusters, data = s_tofu, show.clust.cent = TRUE,
             ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,
             pointsize = 0.5, outlier.color = "darkred", geom = "point") +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Resultados clustering K-means Tofu")
```




```
set.seed(101)
hc_euclidea_av <- hclust(d = dist(x = s_tofu, method = "euclidean"),
                         method = "average")
fviz_dend(x = hc_euclidea_av, k = 4, cex = 0.5,
          k_colors = c("red","blue","green","magenta"),color_labels_by_k = T,
          lwd = 0.2,type = "r",label_cols = rainbow(nrow(df_tofu)),
          rect_lty = "lightblue") +
  geom_hline(yintercept = 3.65, linetype = "dashed") +
  labs(title = "Herarchical clustering Tofu",
       subtitle = "Distancia euclidea, Average, k=4")
```

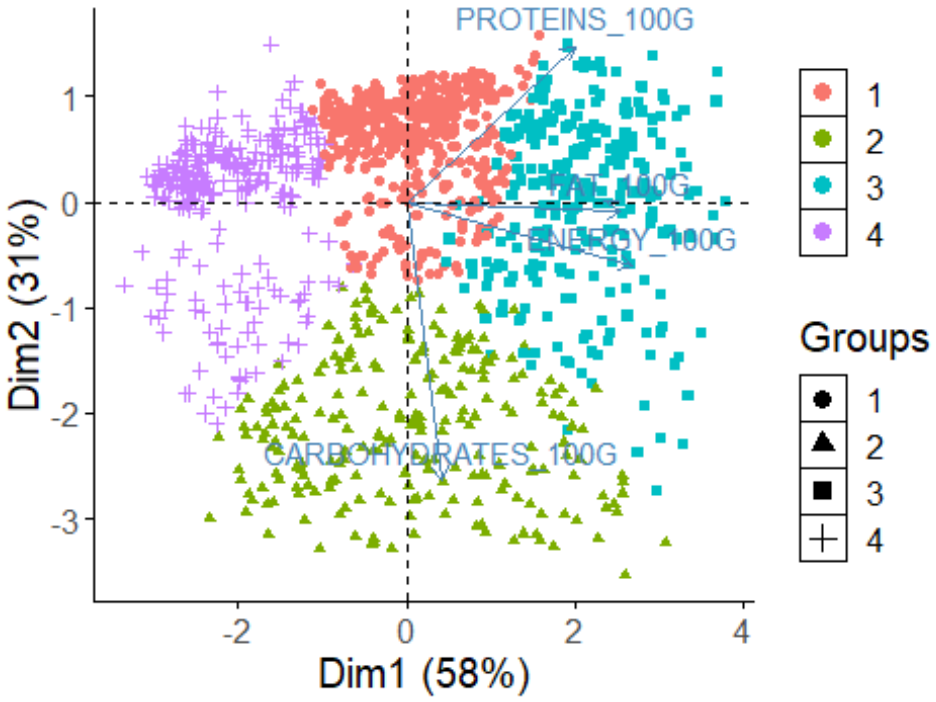



```
pam.res <- pam(s_tofu, 4)
# Visualización
fviz_cluster(pam.res, geom = "point", ellipse.type = "norm",
             show.clust.cent = TRUE,star.plot = TRUE)+
  labs(title = "Resultados clustering K-means Superpuestos Tofu")+ theme_bw()
```



Biplot PCA y K-Means para medir representatividad tofu

```
# PCA
pca <- prcomp(df_tofu[,-1:-2], scale=TRUE)
df_tofu.pca <- pca$x
# Cluster over the three first PCA dimensions
kc <- kmeans(df_tofu.pca[,1:3], 4)
fviz_pca_biplot(pca, label="var", habillage=as.factor(kc$cluster)) +
  labs(color=NULL) + ggtitle("") +
  theme(text = element_text(size = 15),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        legend.key = element_rect(fill = "white"))
```



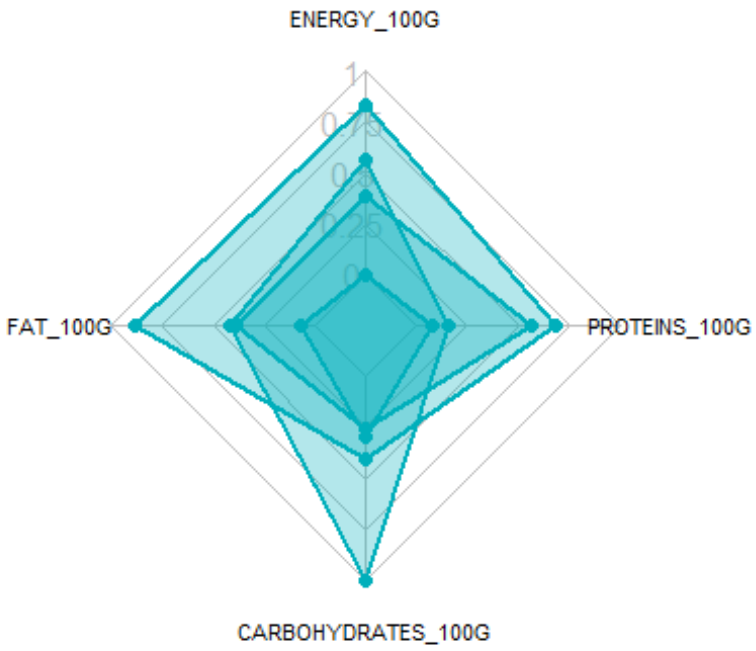
El gráfico previo ilustra los productos en el plano, se observa primero como las variables ENERGY y FAT se encuentran altamente correlacionadas con el eje 1, por otra parte, CARBOHYDRATES esta correlacionada con el eje 2, finalmente PROTEINS esta explicada parcialmente por el eje 1 y 2. Productos cercanos a la dirección de los vectores indican valores altos en esas variables, productos opuestos a la dirección de los vectores indican bajos valores en esas variables. En este sentido, se puede identificar lo siguiente:

- Grupo 1. Productos de TOFU con valores intermedios en energía y grasas, valores bajos en carbohidratos ligeramente altos en Proteínas.
- Grupo 2. Productos con valores altos en Carbohidratos y bajos en proteinas, con valores ligeramente intermedios en grasas y energías..
- Grupo 3. Productos con valores altos en proteínas, grasas y energías pero con valores bajos en carbohidratos.
- Grupo 4. Productos de TOFU con valore bajos en carbohidratos, grasa y energía y ligeramente bajos en proteínas.

A continuación se muestran estos resultados de otra manera mediante gráficos de radar:

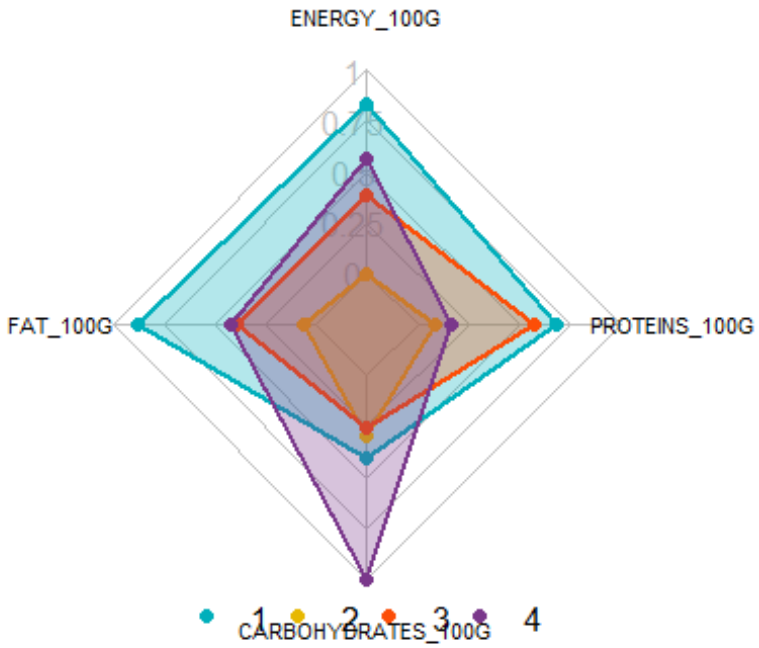
```
library(ggplot2)

# Obtener los centroides de cada clúster
centroids <- as.data.frame(km_clusters$centers)


# Obtener el valor mínimo y máximo en el dataframe
min_value <- min(centroids, na.rm = TRUE)
```

```
max_value <- max(centroids, na.rm = TRUE)

# Escalar los datos entre 0 y 1
scaled_df <- (centroids - min_value) / (max_value - min_value)

library(fmsb)


# Define the variable ranges: maximum and minimum
max_min <- data.frame(
  ENERGY_100G = c(1, 0), FAT_100G = c(1, 0), CARBOHYDRATES_100G = c(1,
0),PROTEINS_100G = c(1, 0)
)
rownames(max_min) <- c("Max", "Min")

# Bind the variable ranges to the data
df <- rbind(max_min, scaled_df)
```

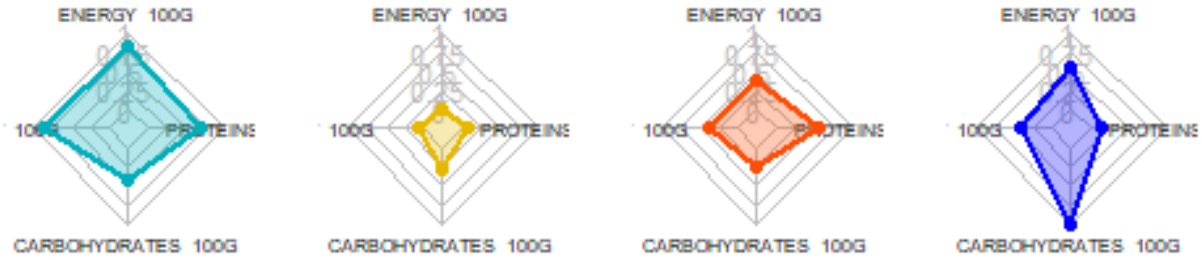



```
par(op)

# Reduce plot margin using par()
op <- par(mar = c(1, 2, 2, 2))
# Create the radar charts
create_beautiful_radarchart(
  data = df, caxislabels = c(0, .25, .5, .75, 1),
  color = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B")
)
```

```
# Add an horizontal legend
legend(
  x = "bottom", legend = rownames(df[-c(1,2),]), horiz = TRUE,
  bty = "n", pch = 20 , col = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B"),
  text.col = "black", cex = 1, pt.cex = 1.5
  )
```




```
par(op)

# Define colors and titles
colors <- c("#00AFBB", "#E7B800", "#FC4E07","blue")
titles <- c("1", "2", "3","4")

# Reduce plot margin using par()
# Split the screen in 3 parts
op <- par(mar = c(1, 1, 1, 1))

par(mfrow = c(1,4))

# Create the radar chart
for(i in 1:4){
  create_beautiful_radarchart(
    data = df[c(1, 2, i+2), ], caxislabels = c(0, .25, .5, .75, 1),
    color = colors[i], title = titles[i]
    )
}
```




```
par(op)
```

Visualizar los productos más cercanos al centroide que representan cada cluster top 10 para el Tofu

```
# Realizar clustering en el dataframe
set.seed(123)
df_top <- df_tofu
km_clusters <- kmeans(x = df_top[, c("ENERGY_100G", "FAT_100G",
"CARBOHYDRATES_100G", "PROTEINS_100G")], centers = 4, nstart = 50)

# Obtener las asignaciones de clúster
cluster_assignments <- km_clusters$cluster

# Agregar las asignaciones de clúster al dataframe
df_top$cluster <- cluster_assignments

# Inicializar una lista para almacenar los productos más cercanos a cada
centroide
closest_products <- vector("list", max(cluster_assignments))

# Encontrar los productos más cercanos a cada centroide
for (cluster in 1:max(cluster_assignments)) {
  cluster_center <- km_clusters$centers[cluster, ]
  distances <- apply(df_top[, c("ENERGY_100G", "FAT_100G",
"CARBOHYDRATES_100G", "PROTEINS_100G")], 1, function(row) {
    sum((row - cluster_center)^2)
  })
  closest_products[[cluster]] <- head(order(distances), 10)
}

# Imprimir los productos más cercanos a cada centroide
for (cluster in 1:max(cluster_assignments)) {
  cat("Cluster", cluster, ":\n")
  print(df_top[closest_products[[cluster]], c("PRODUCT_NAME", "ENERGY_100G",
"FAT_100G", "CARBOHYDRATES_100G", "PROTEINS_100G")])
  cat("\n")
}
```

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
385	MEDIUM FIRM TOFU	295	3.529412	2.352941	7.058824
723	TOFU	295	3.529412	2.352941	7.058824
730	HOUSE FOODS TOFU REGULAR	295	3.529412	2.352941	7.058824
735	TOFU	295	3.529412	2.352941	7.058824
749	TOFU MEDIUM FIRM	295	3.529412	2.352941	7.058824
1429	ORGANIC SILKEN TOFU	295	3.529412	2.352941	7.058824
1432	PULMUONE TOFU	295	3.529412	2.352941	7.058824
736	SOFT TOFU	295	3.529412	2.352941	5.882353
1577	FIRM TOFU	295	3.529412	2.352941	8.235294
1796	TOFU	295	3.529412	2.352941	8.235294

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
6719	TOFU FUME	707	10.0000	3.20	16.0000
2014	BIO-TOFU	707	9.5000	2.00	18.0000
2346	BIO TOFU GERAUCHERT	707	9.5000	2.00	18.0000
5122	RAUCHER-TOFU	707	9.5000	2.00	18.0000
5084	BIO-TOFU GERAUCHERT	708	9.7143	2.00	18.2857
1800	TOFU YU, TERIYAKI TOFU SUSHI	703	10.1800	7.96	15.4900
4832	BIO TOFU GERAUCHERT	704	9.5000	2.00	18.0000
5070	RAUCHER TOFU SCHNITTFEST	704	9.5000	2.00	18.0000
5100	RAUCHER-TOFU KRAFTIG	704	9.5000	2.00	18.0000
2618	TOFU NATURE	705	9.8000	0.70	18.0000

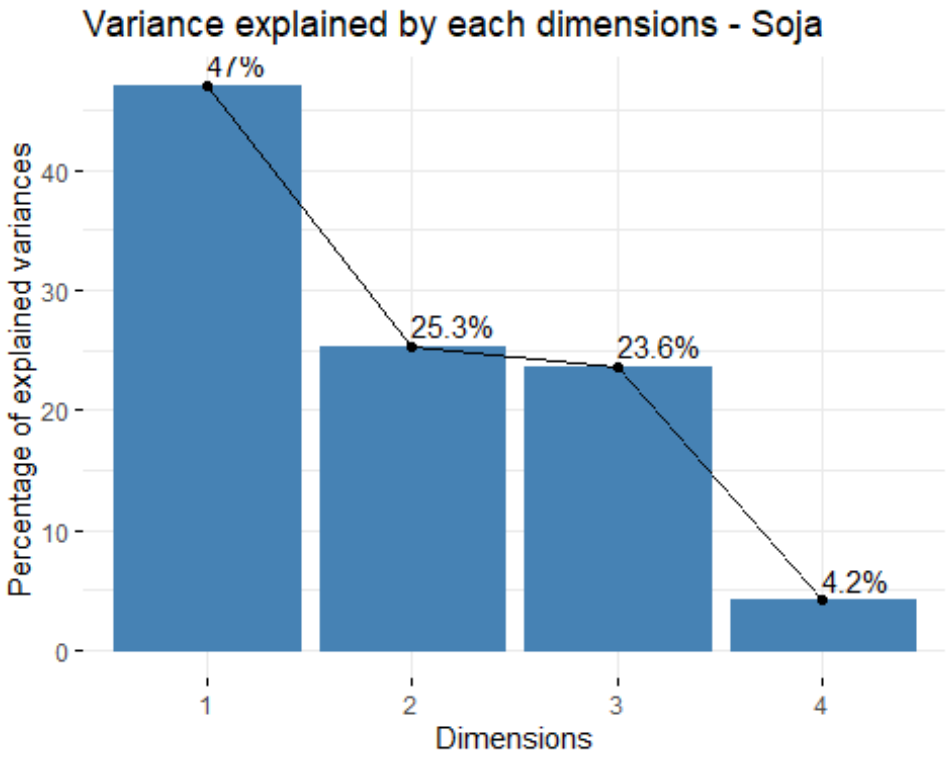
	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
7389	COTOLETTA DI TOFU	950	14.00	12.00	12.00
7571	COTOLETTA DI TOFU E SEITAN	946	11.79	10.80	15.60
7562	COTOLETTA DI TOFU E SEITAN	946	11.70	10.80	15.60
8407	SALCHICHAS TOFU FRANKFURT	950	13.47	5.44	20.61
8698	HAPPYBURGER TOFU CHAMPIGNON	954	14.00	13.00	11.00
6751	QUINOA TOFU BALLS	950	14.00	12.00	9.00
8143	VEGEBURGER DE TOFU Y CHAMPINONES	954	13.70	13.10	10.70
8161	VEGEBURGER DE TOFU Y CHAMPINONES	954	13.70	13.10	10.70
8686	VEGBURGUER TOFU Y CHAMPINONES	954	13.70	13.10	10.70
5446	FISH TOFU	946	15.40	11.20	10.60

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
8662	TOFU CON SEMILLA DE AMAPOLA	527	6.80	3.3	12.20
695	ORGANIC FIRM TOFU	531	6.33	3.8	12.66
807	O ORGANICS, ORGANIC EXTRA FIRM CUBED TOFU	531	6.33	3.8	12.66
8399	ORGANIC TOFU JAPONES	527	8.00	0.7	12.26
4530	TOFU TERIYAKI	531	7.00	3.0	13.00
4313	FETO TOFU FERMENTE AUX HERBES	527	6.40	1.9	15.00
4608	BIO TOFU	527	7.90	0.0	13.00
4611	BIO TOFU NATURE	527	7.90	0.0	13.00
4697	TOFU NATUR	527	7.90	0.0	13.00
6437	BIO TOFU AL NATURALE	527	7.90	0.0	13.00

03. Soja

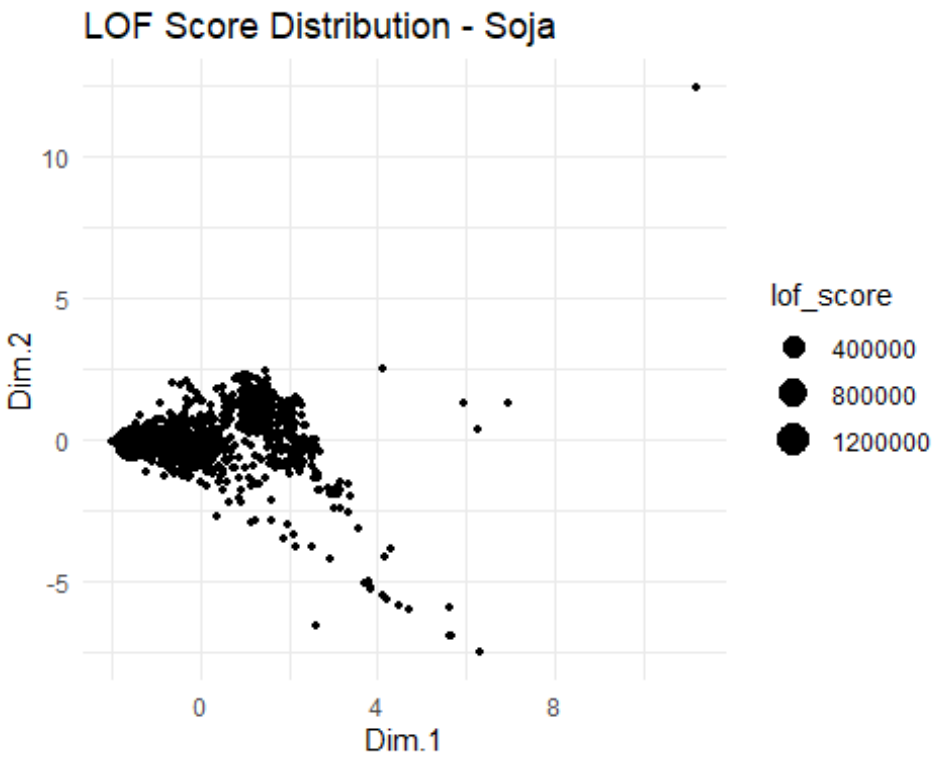
Los resultados para este producto son los siguientes

```
fviz_eig(soja_pca, ncp = 6, addlabels = T, main = "Variance explained by each dimensions - Soja")
```

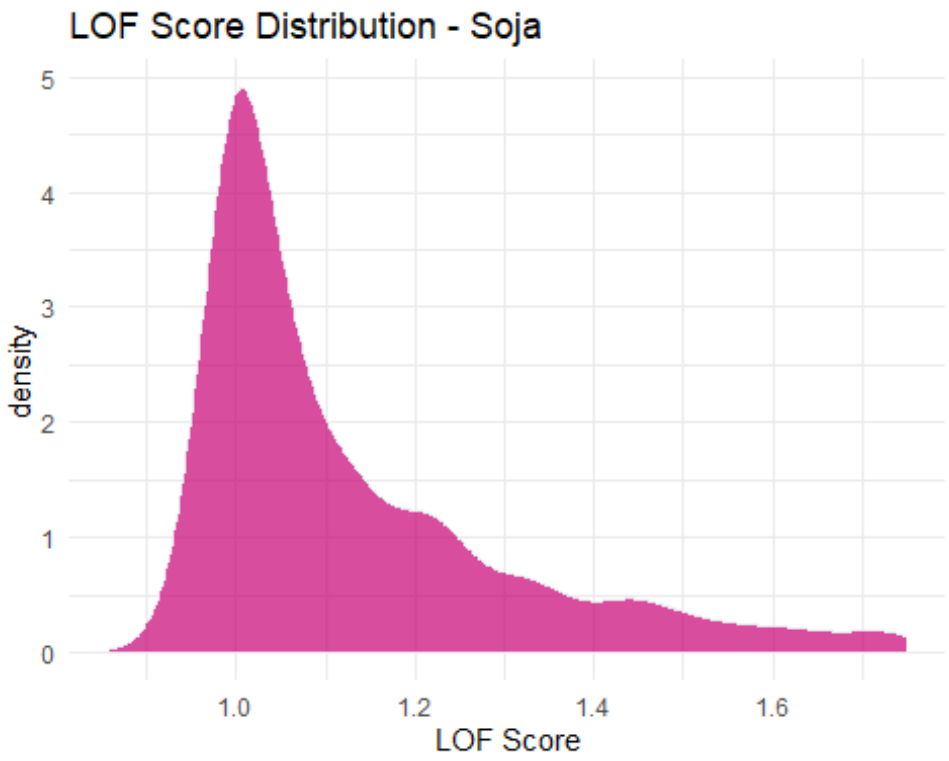


Técnica LOF - Soja

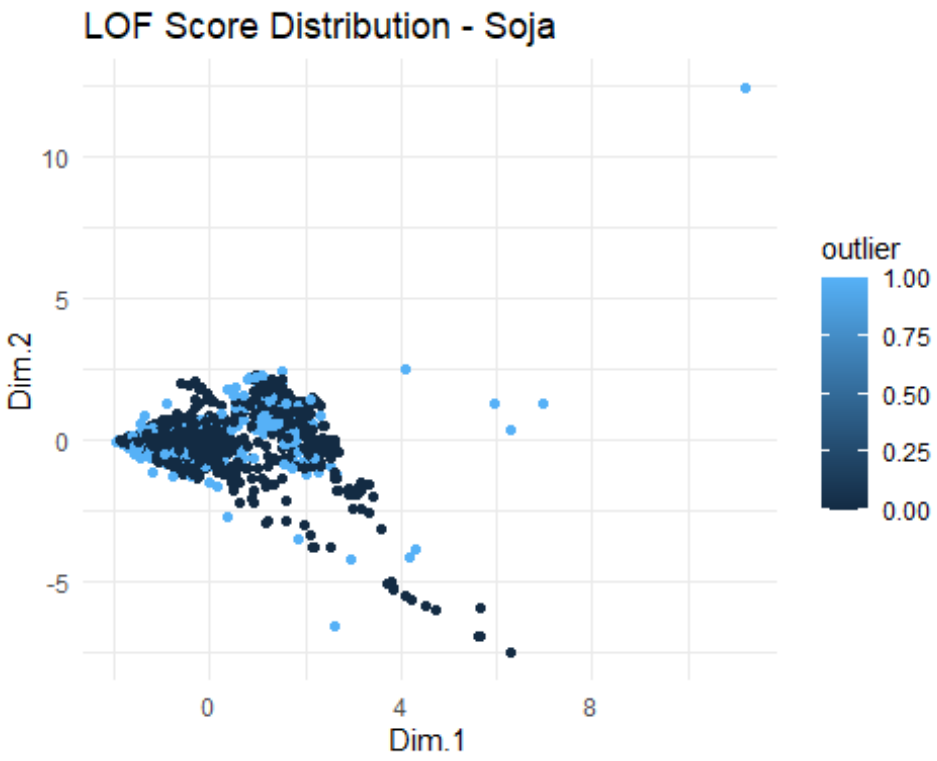
```
library(ggthemes)
soja_a <- data.frame(soja_pca$ind$coord[,1:3])
soja_b <- cbind(soja_a, lof_score = soja_lof)
#soja_b <- cbind(soja_a, fraud = soja_clean$, lof_score = soja_clean$lof)

soja_lof_visual <- ggplot(soja_b, aes(x=Dim.1 ,y=Dim.2)) +
    geom_point(aes(size=lof_score)) +
  ggtitle("LOF Score Distribution - Soja")+
    theme_minimal()

soja_lof_visual
```




```
summary(soja_b)

##      Dim.1              Dim.2              Dim.3             lof_score
##  Min.   :-1.9743   Min.   :-7.50403   Min.   :-3.59918   Min.   :0.8579
##  1st Qu.:-1.2351   1st Qu.:-0.34531   1st Qu.:-0.35626   1st Qu.:1.0058
##  Median :-0.3107   Median :-0.03605   Median : 0.01204   Median :1.0928
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000   Mean   :   Inf
##  3rd Qu.: 1.1951   3rd Qu.: 0.38722   3rd Qu.: 0.32017   3rd Qu.:1.3530
##  Max.   :11.2007   Max.   :12.42671   Max.   :18.73336   Max.   :   Inf

soja_b %>%
  filter(lof_score <= 1.75) %>%
  ggplot( aes(x=lof_score)) +
    geom_density( color="#e9ecef", fill = "#c90076", alpha=0.7) +
    scale_fill_manual(values="#8fce00") +
    xlab("LOF Score")+
  ggtitle("LOF Score Distribution - Soja")+
    theme_minimal() +
    labs(fill="")
```




```
quantile(soja_b$lof_score, probs = c(0, 0.8))

##       0%      80%
## 0.857934 1.485425

soja_b <- soja_b %>%
  mutate(outlier = ifelse(lof_score > 1.485425, 1, 0))

soja_lof_visual_b <- ggplot(soja_b, aes(x=Dim.1 ,y=Dim.2, color=outlier)) +
    geom_point() +
  ggtitle("LOF Score Distribution - Soja")+
    theme_minimal()

soja_lof_visual_b
```

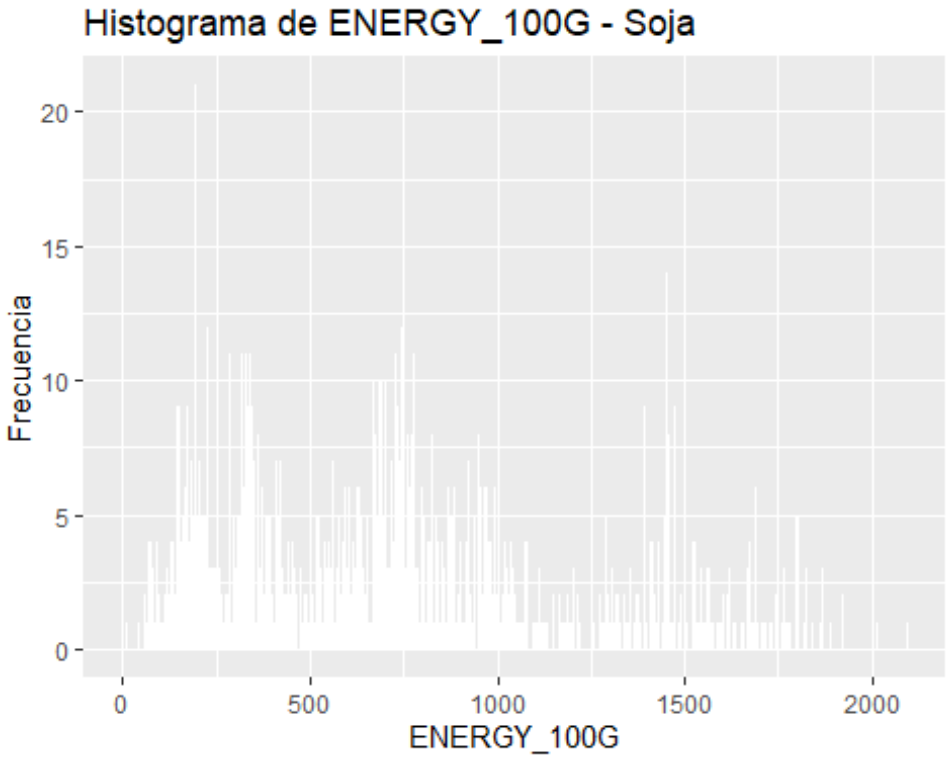


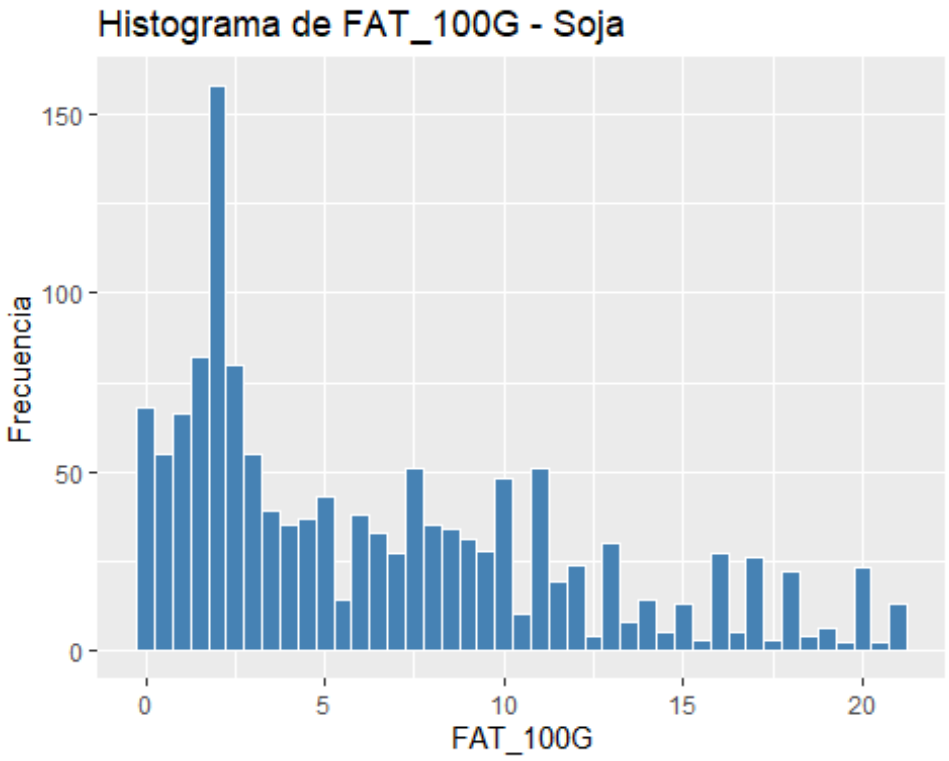

```
outliers <- soja_b[soja_b$outlier==1,]
nooutliers <- soja_b[soja_b$outlier==0,]

df_soja <- df_soja[soja_lof < 1.485425,]
df_soja_outliers <- df_soja[soja_lof >= 1.485425,]
```

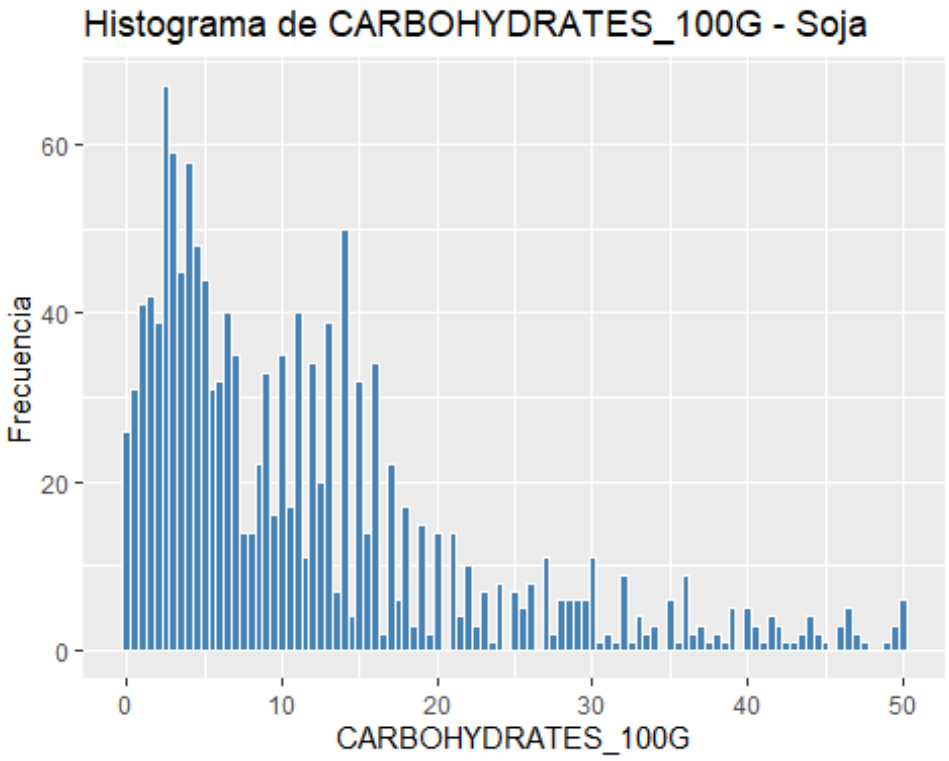
Eliminar valores atipicos univariantes

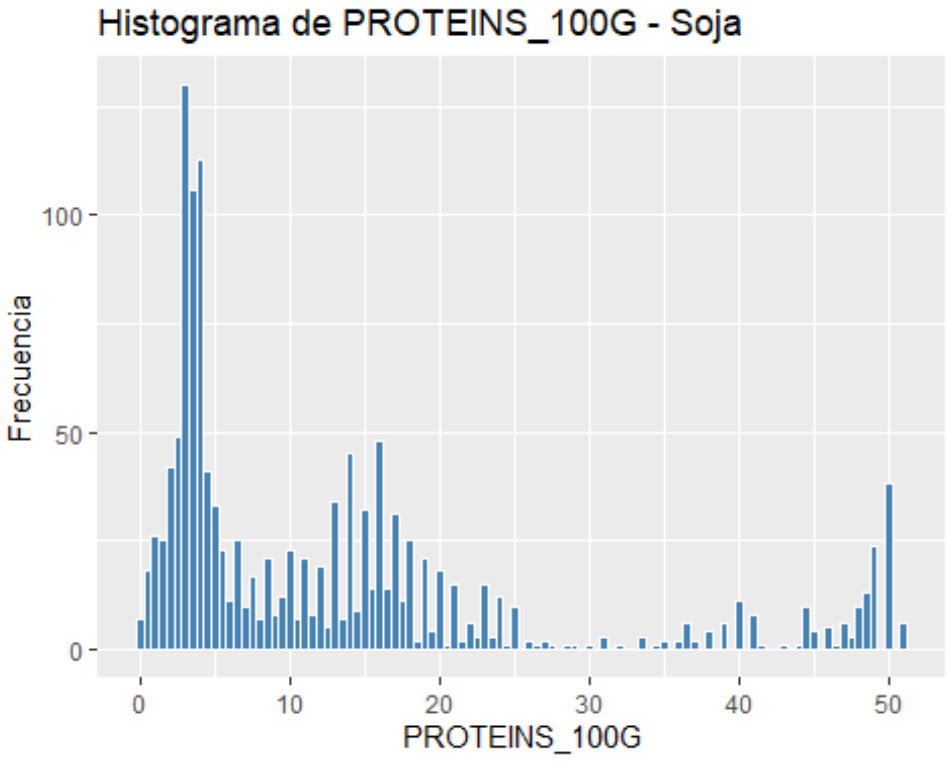
```
remove_outliers <- function(data, column, sd_threshold = 2) {
  data[abs(scale(data[[column]])) < sd_threshold, ]
}

columns_to_check <- 3:ncol(df_soja)
for (column in columns_to_check) {
  df_soja <- remove_outliers(df_soja, column)
}

# Obtener las columnas cuantitativas del dataframe
columnas_cuantitativas <- sapply(df_soja, is.numeric)

# Crear un histograma para cada columna cuantitativa
for (columna in names(df_soja[columnas_cuantitativas])) {
  plot_data <- df_soja[, columna]
  p <- ggplot(data.frame(x = plot_data), aes(x)) +
    geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +
    labs(title = paste("Histograma de", columna,"- Soja"),
         x = columna,
```

```
        y = "Frecuencia")

  print(p)
}
```









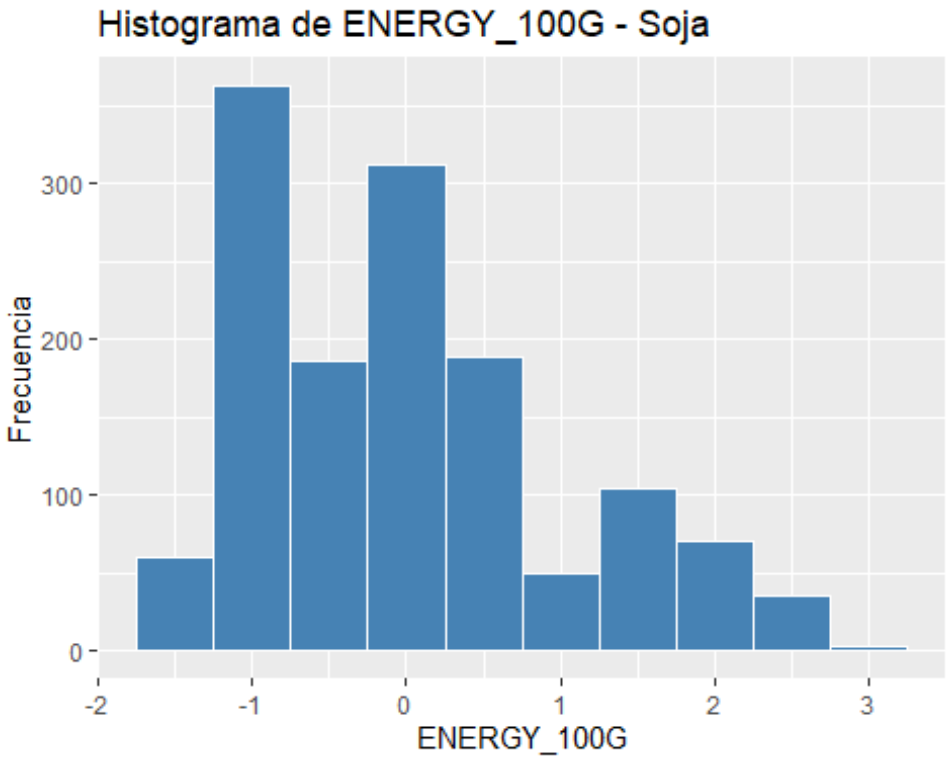

```
summary(as.data.frame(scale(df_soja[,-1:-2])))
```

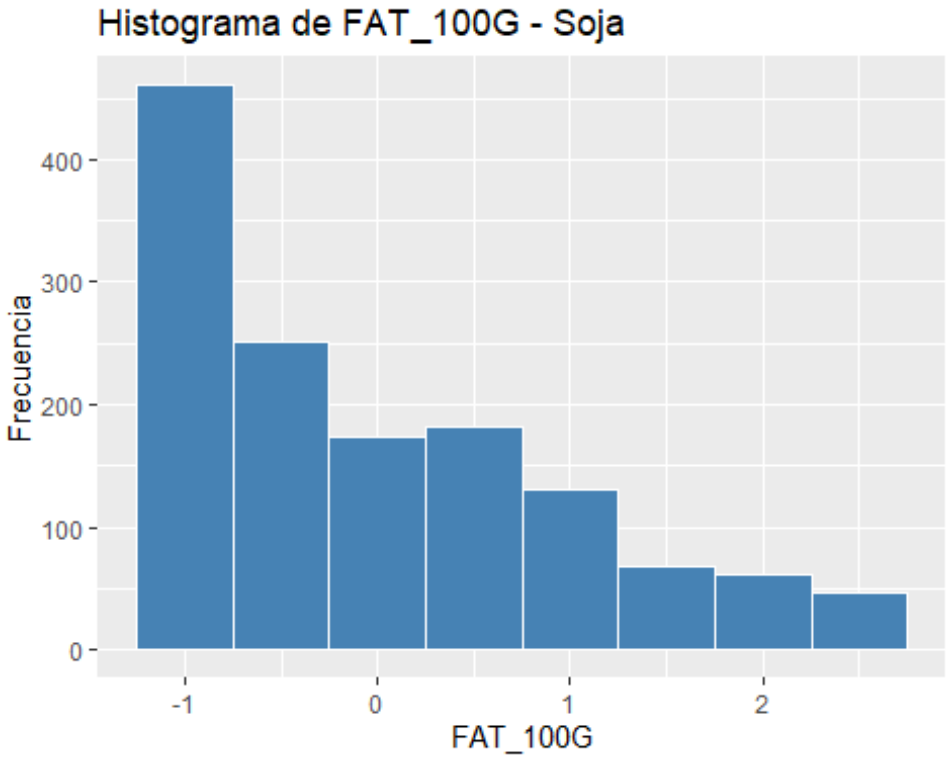
```
##   ENERGY_100G          FAT_100G       CARBOHYDRATES_100G PROTEINS_100G
##  Min.   :-1.53946   Min.   :-1.1866   Min.   :-1.0813    Min.   :-0.9730
```

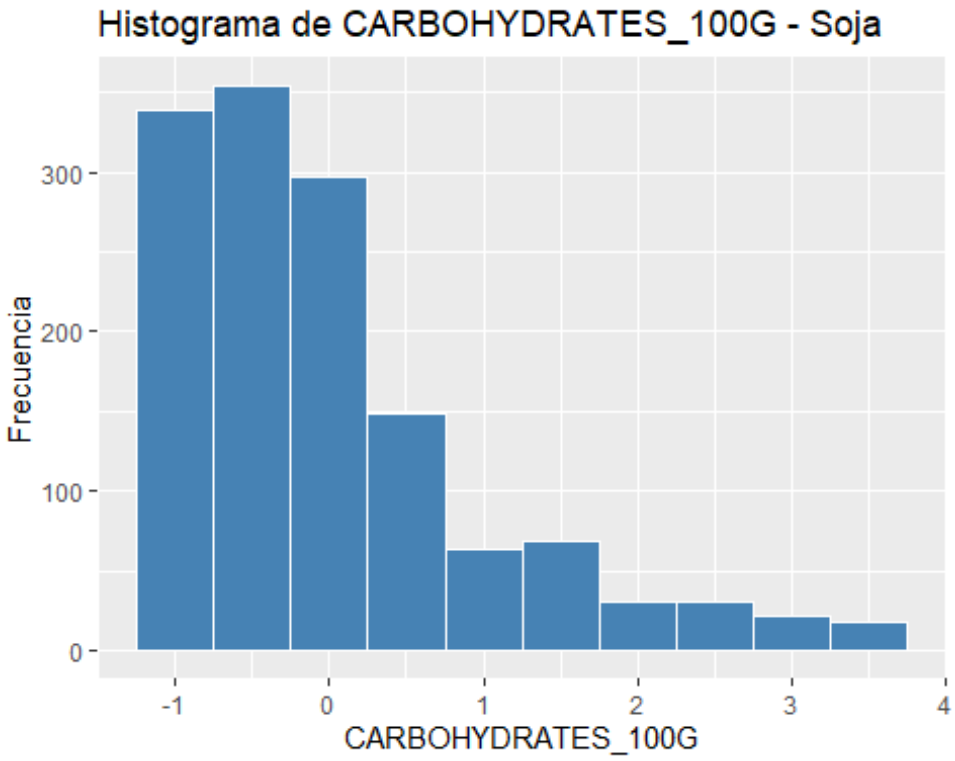
```
##  1st Qu.:-0.82857   1st Qu.:-0.8203   1st Qu.:-0.7347    1st Qu.:-0.7118
##  Median :-0.08933   Median :-0.2708   Median :-0.2663    Median :-0.3708
##  Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000    Mean   : 0.0000
##  3rd Qu.: 0.54087   3rd Qu.: 0.6450   3rd Qu.: 0.3613    3rd Qu.: 0.2605
##  Max.   : 3.02243   Max.   : 2.6596   Max.   : 3.6024    Max.   : 2.7275
```

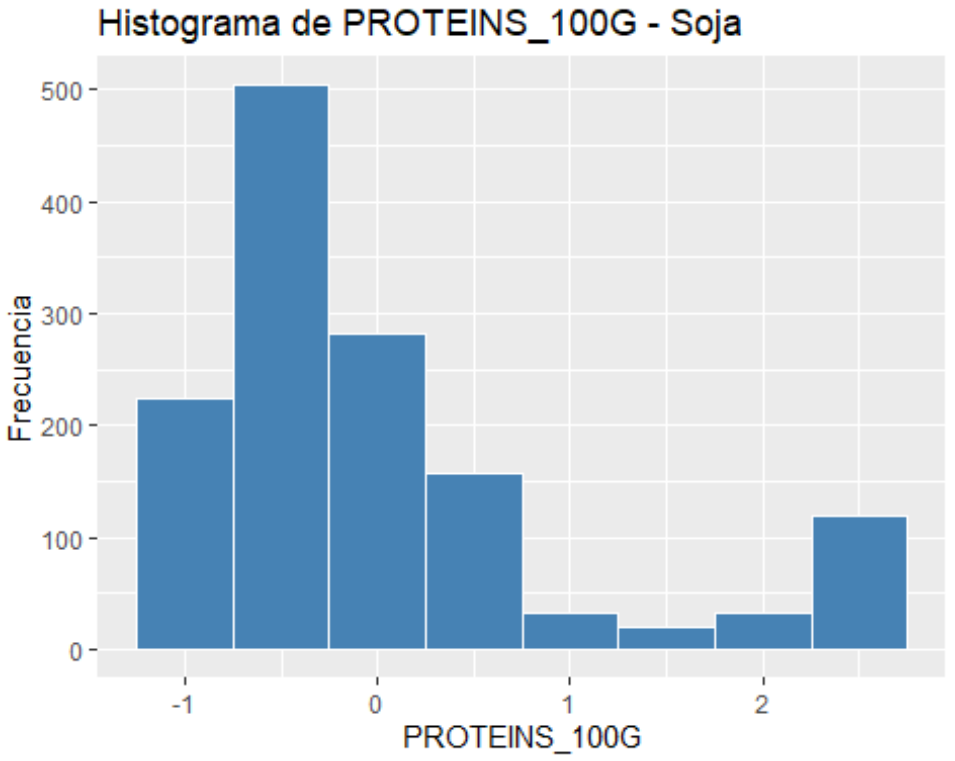
Datos normalizados para el soja

```
s_soja <- as.data.frame(scale(df_soja[,-1:-2]))
# Obtener las columnas cuantitativas del dataframe
columnas_cuantitativas <- sapply(s_soja, is.numeric)

# Crear un histograma para cada columna cuantitativa
for (columna in names(s_soja[columnas_cuantitativas])) {
  plot_data <- s_soja[, columna]
  p <- ggplot(data.frame(x = plot_data), aes(x)) +
    geom_histogram(binwidth = 0.5, fill = "steelblue", color = "white") +
    labs(title = paste("Histograma de", columna,"- Soja"),
         x = columna,
         y = "Frecuencia")

  print(p)
}
```


Histograma de ENERGY_100G - Soja
Frecuencia
300
200
100
0
-2
-1
0
1
2
3
ENERGY_100G

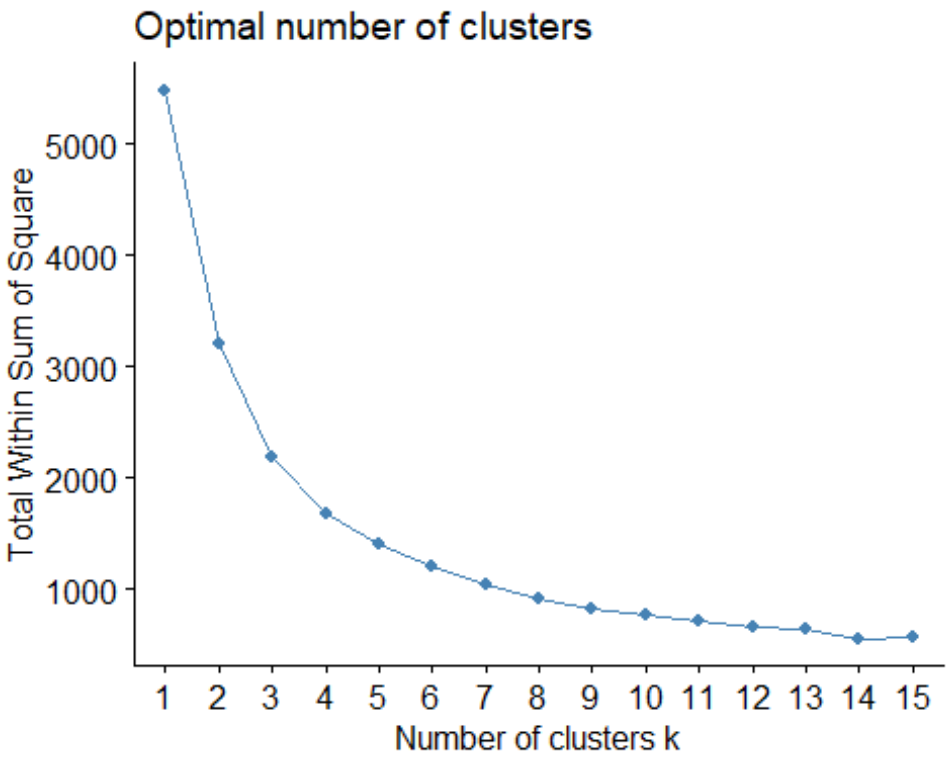

Histograma de FAT_100G - Soja
Frecuencia
400
300
200
100
0
-1
0
1
2
FAT_100G



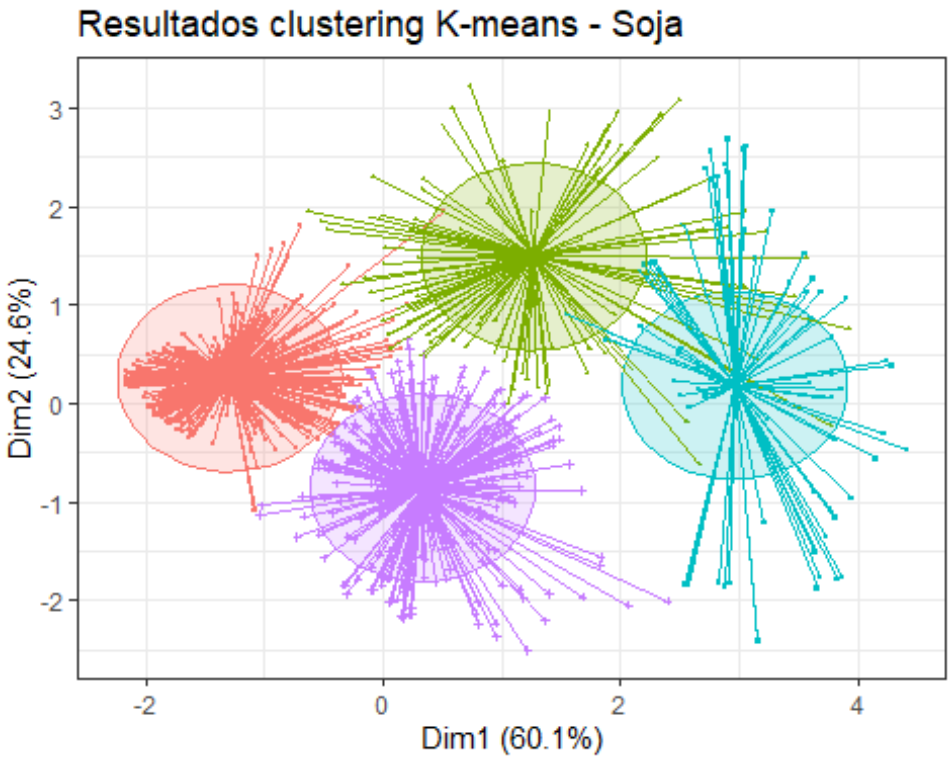



```
s_soja <- scale(df_soja[,-1:-2])

# total de cluster óptimos
elbow <- fviz_nbclust(x = s_soja, FUNcluster = kmeans, method = "wss", k.max
```

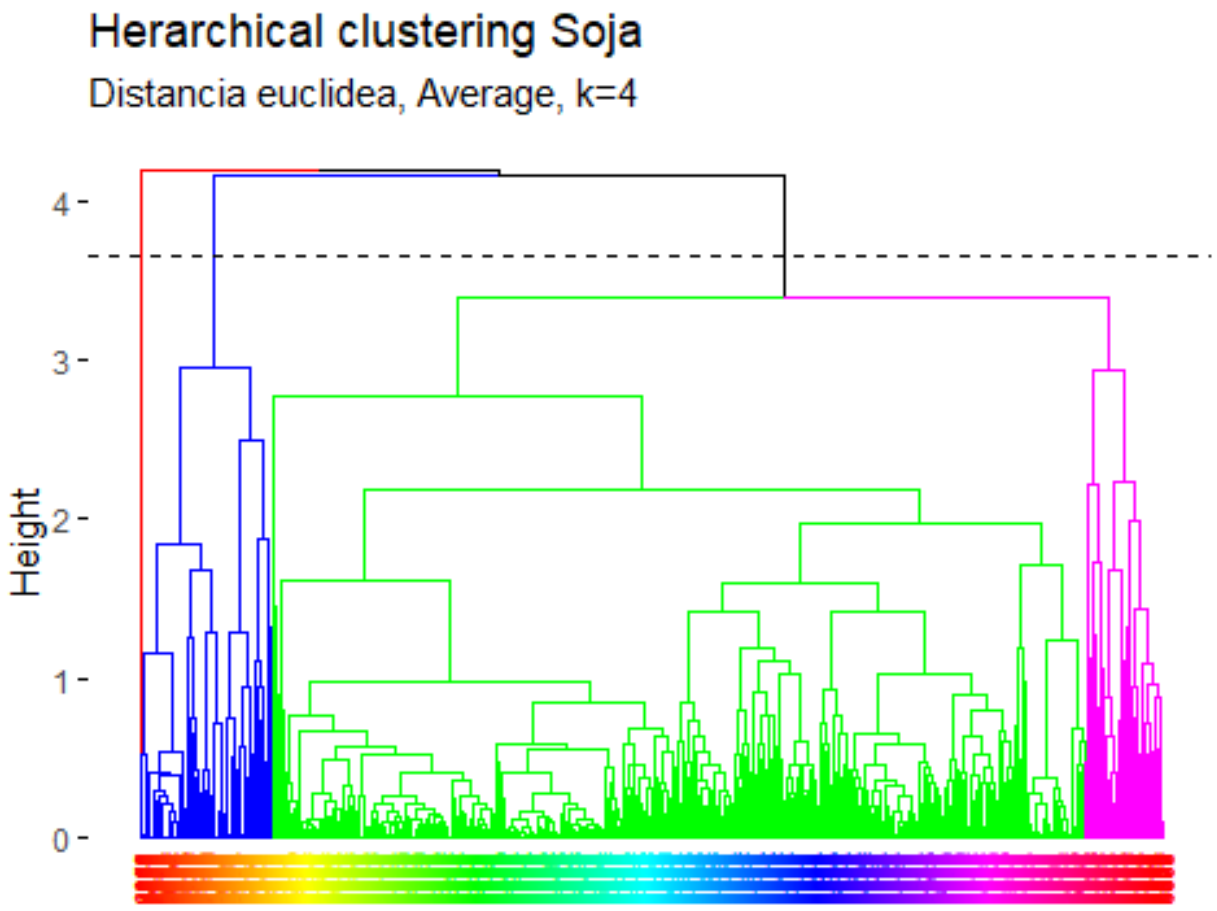
```
= 15,
              diss = get_dist(s_soja, method = "euclidean"), nstart = 25)
print(elbow)
```



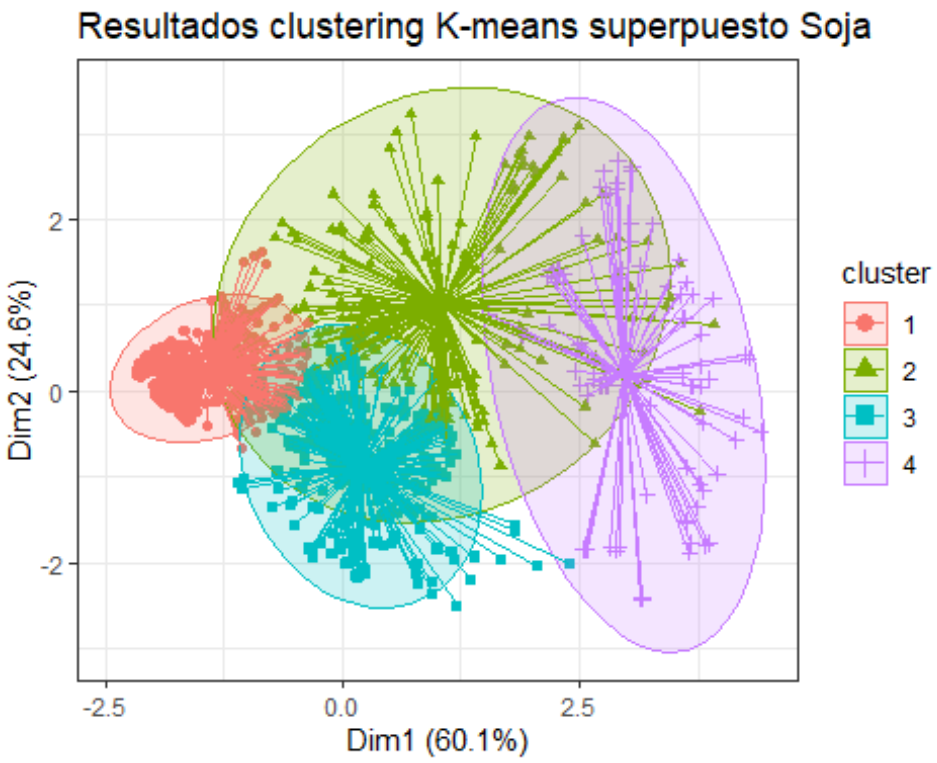

```
set.seed(123)
km_clusters <- kmeans(x = s_soja, centers = 4, nstart = 50)

fviz_cluster(object = km_clusters, data = s_soja, show.clust.cent = TRUE,
             ellipse.type = "euclid", star.plot = TRUE, repel = TRUE,
             pointsize = 0.5, outlier.color = "darkred", geom = "point") +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Resultados clustering K-means - Soja")
```




```
set.seed(101)
hc_euclidea_av <- hclust(d = dist(x = s_soja, method = "euclidean"),
                         method = "average")
fviz_dend(x = hc_euclidea_av, k = 4, cex = 0.5,
          k_colors = c("red","blue","green","magenta"),color_labels_by_k = T,
          lwd = 0.2,type = "r",label_cols = rainbow(nrow(df_soja)),
          rect_lty = "lightblue") +
  geom_hline(yintercept = 3.65, linetype = "dashed") +
  labs(title = "Herarchical clustering Soja",
       subtitle = "Distancia euclidea, Average, k=4")
```

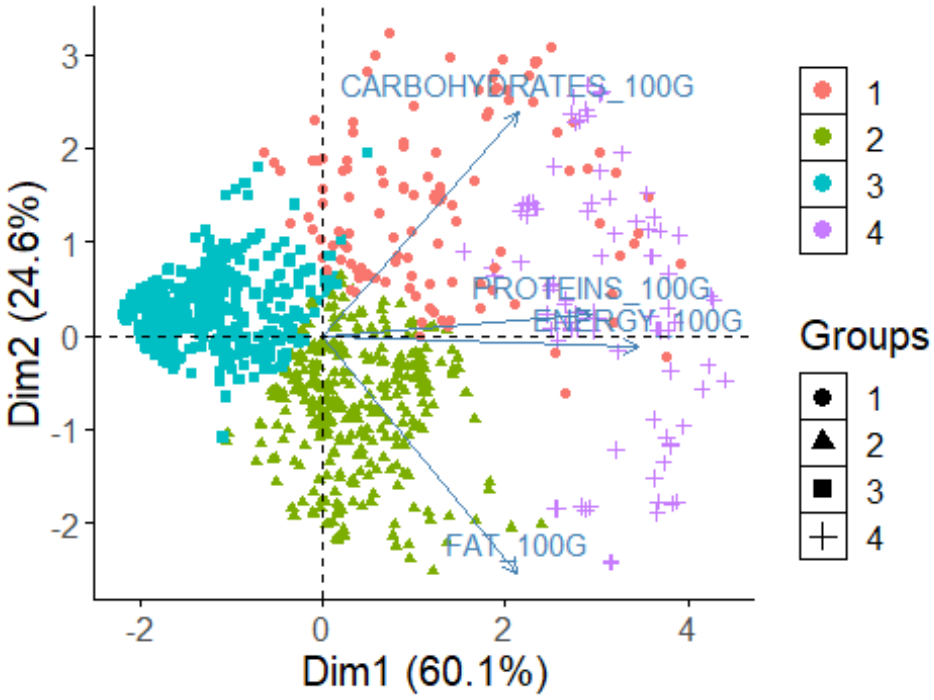



```
pam.res <- pam(s_soja, 4)
# Visualización
fviz_cluster(pam.res, geom = "point", ellipse.type = "norm",
             show.clust.cent = TRUE,star.plot = TRUE)+
  labs(title = "Resultados clustering K-means superpuesto Soja")+ theme_bw()
```



Biplot PCA y K-Means para medir representatividad soja

```
# PCA
pca <- prcomp(df_soja[,-1:-2], scale=TRUE)
df_soja.pca <- pca$x
# Cluster over the three first PCA dimensions
kc <- kmeans(df_soja.pca[,1:3], 4)
fviz_pca_biplot(pca, label="var", habillage=as.factor(kc$cluster)) +
  labs(color=NULL) + ggtitle("") +
  theme(text = element_text(size = 15),
        panel.background = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.line = element_line(colour = "black"),
        legend.key = element_rect(fill = "white"))
```



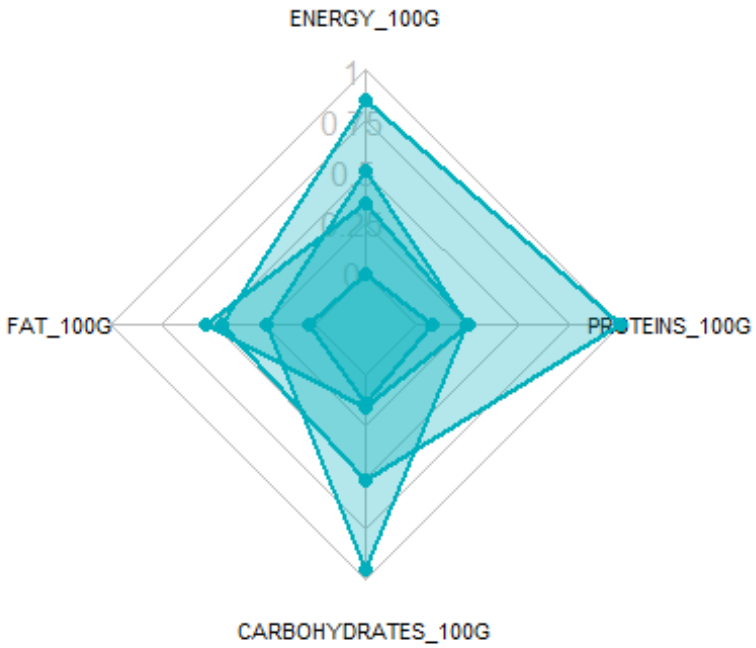
Aunque los dos ejes explican el 85% de variabilidad total con los 2 primeros autovalores, en este cuadrante todavía los cluster no se observan bien definidos, dado que este producto se encuentra aún muy contaminado por elementos que no son objeto de este estudio, sin embargo, se intentará explicar.

El gráfico previo ilustra los productos en el plano, se observa primero como las variables ENERGY y FAT se encuentran altamente correlacionadas con el eje 1, por otra parte, CARBOHYDRATES y PROTEINS estan explicadas parcialmente por el eje 1 y 2. Productos cercanos a la dirección de los vectores indican valores altos en esas variables, productos opuestos a la dirección de los vectores indican bajos valores en esas variables. En este sentido, se puede identificar lo siguiente:

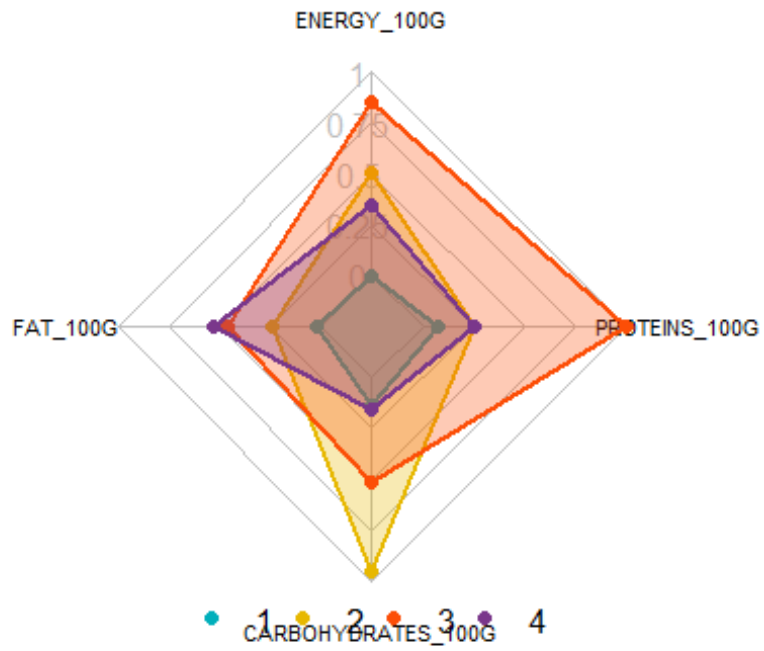
- Grupo 1. Productos de soja en el cuadrante superior derecho, con valores altos en carbohidratos, bajos en grasa..
- Grupo 2. Productos con valores altos en Grasas y bajos en Carbohidratos y valores intermedios en proteínas y energía.
- Grupo 3. Productos con valores bajos en proteínas, grasas y energías pero con valores intermedios en grasas y carbohidratos.
- Grupo 4. Productos de soja con valore altos grasa y energía pero con valores altos y bajos en grasas y carbohidratos..

A continuación se muestran estos resultados de otra manera mediante gráficos de radar:

```r
library(ggplot2)

# Obtener los centroides de cada clúster
centroids <- as.data.frame(km_clusters$centers)


# Obtener el valor mínimo y máximo en el dataframe
min_value <- min(centroids, na.rm = TRUE)
max_value <- max(centroids, na.rm = TRUE)

# Escalar los datos entre 0 y 1
scaled_df <- (centroids - min_value) / (max_value - min_value)

library(fmsb)


# Define the variable ranges: maximum and minimum
max_min <- data.frame(
  ENERGY_100G = c(1, 0), FAT_100G = c(1, 0), CARBOHYDRATES_100G = c(1,
0),PROTEINS_100G = c(1, 0)
)
rownames(max_min) <- c("Max", "Min")

# Bind the variable ranges to the data
df <- rbind(max_min, scaled_df)
```

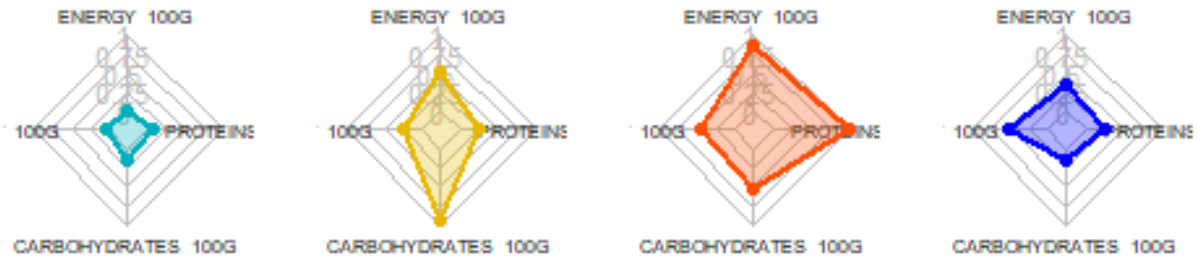


```
par(op)

# Reduce plot margin using par()
op <- par(mar = c(1, 2, 2, 2))
# Create the radar charts
create_beautiful_radarchart(
  data = df, caxislabels = c(0, .25, .5, .75, 1),
  color = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B")
)
# Add an horizontal legend
legend(
  x = "bottom", legend = rownames(df[-c(1,2),]), horiz = TRUE,
  bty = "n", pch = 20 , col = c("#00AFBB", "#E7B800", "#FC4E07","#7A378B"),
  text.col = "black", cex = 1, pt.cex = 1.5
  )
```




```
par(op)

# Define colors and titles
colors <- c("#00AFBB", "#E7B800", "#FC4E07","blue")
titles <- c("1", "2", "3","4")

# Reduce plot margin using par()
# Split the screen in 3 parts
op <- par(mar = c(1, 1, 1, 1))

par(mfrow = c(1,4))
```

```r
# Create the radar chart
for(i in 1:4){
  create_beautiful_radarchart(
    data = df[c(1, 2, i+2), ], caxislabels = c(0, .25, .5, .75, 1),
    color = colors[i], title = titles[i]
    )
}
```




```r
par(op)
```

Visualizar los productos más cercanos al centroide que representan cada cluster top 10 para la soja

```r
# Realizar clustering en el dataframe
set.seed(123)
df_top <- df_soja
km_clusters <- kmeans(x = df_top[, c("ENERGY_100G", "FAT_100G",
"CARBOHYDRATES_100G", "PROTEINS_100G")], centers = 4, nstart = 50)

# Obtener las asignaciones de clúster
cluster_assignments <- km_clusters$cluster

# Agregar las asignaciones de clúster al dataframe
df_top$cluster <- cluster_assignments

# Inicializar una lista para almacenar los productos más cercanos a cada
centroide
closest_products <- vector("list", max(cluster_assignments))

# Encontrar los productos más cercanos a cada centroide
for (cluster in 1:max(cluster_assignments)) {
  cluster_center <- km_clusters$centers[cluster, ]
  distances <- apply(df_top[, c("ENERGY_100G", "FAT_100G",
"CARBOHYDRATES_100G", "PROTEINS_100G")], 1, function(row) {
    sum((row - cluster_center)^2)
  })
  closest_products[[cluster]] <- head(order(distances), 10)
}

# Imprimir los productos más cercanos a cada centroide
```

```
for (cluster in 1:max(cluster_assignments)) {
  cat("Cluster", cluster, ":\n")
  print(df_top[closest_products[[cluster]], c("PRODUCT_NAME", "ENERGY_100G",
"FAT_100G", "CARBOHYDRATES_100G", "PROTEINS_100G")])
  cat("\n")
}
```

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
3165	SOJA FERMENTADA	264	2.3	6.4	4.1
2895	BOULETTES DE SOJA ET TOMATES	262	1.5	6.3	4.0
6816	SOJA	264	0.0	7.0	8.7
4977	16G PROTEIN OHNE SOJA	268	0.3	8.8	6.4
3062	SO SOJA ! BANANE-PASSION	259	2.0	7.0	3.6
4772	GULASCH VEGAN MIT SOJA	259	1.1	6.5	4.0
4338	SHOYU - SOJASOSSE JAPANISCHE ART	267	0.2	5.2	10.0
4467	SHOYU SOJASOSSE AUS JAPAN	259	0.1	6.2	8.8
2894	BOUCHEES SOJA ET LEGUMES	272	2.0	5.2	4.5
2896	BOUCHEES DE SOJA ET LEGUMES	272	2.0	5.2	4.5

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
5624	LA PANEE SOJA ET BLE	981	11.66667	17.55556	12.22222
6864	LA PANEE SOJA ET BLE	981	11.66667	17.55556	12.22222
6689	GARDEN GOURMET LA PANEE SOJA ET BLE 180G	984	11.70000	17.60000	12.30000
6701	GARDEN GOURMET LA PANEE SOJA ET BLE FORMAT FAMILIAL 360G	984	11.70000	17.60000	12.30000
3231	BURGER VEGETARIEN STEAK SOJA ET MOZZARELLA	980	10.00000	21.00000	13.00000
2496	GRILL VEGETAL - NUGGETS SOJA & BLE	975	11.00000	14.00000	17.00000
6441	NUGGETS SOJA & BLE	975	11.00000	14.00000	17.00000
5718	BURGERS DE LEGUMES AU SOJA BELGE	983	14.00000	9.20000	15.00000
3390	NUGGETS SOJA ET BLE	983	8.00000	25.00000	13.00000
15	ESCALOPES PANEES DE SOJA	992	11.00000	18.00000	14.00000

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
2949	PROTEINES DE SOJA GROSSES	1544	8.0	14.0	50.0
3254	PROTEINES DE SOJA GROS MORCEAUX	1540	8.0	14.0	50.0
3256	PROTEINE DE SOJA PETITS MORCEAUX	1540	8.0	14.0	50.0
4057	PROTEINE DE SOJA	1540	8.0	14.0	50.0
8051	SOJA FINA TEXTURIZADA	1531	6.0	32.0	46.0
4624	SOJASCHNETZEL	1561	9.0	16.0	50.0
4868	PLANET NATURE SOJA FLEISCHSCHNETZEL ART	1561	9.0	16.0	50.0
9251	BIO SOJA GRANULAT	1561	9.0	16.0	50.0
3364	SOJA VERT	1536	18.1	6.3	33.7
8560	FIDEOS DE SOJA BIOLOGICOS	1527	5.0	19.9	26.8

	PRODUCT_NAME <chr>	ENERGY_100G <dbl>	FAT_100G <dbl>	CARBOHYDRATES_100G <dbl>	PROTEINS_100G <dbl>
2513	CROQ' SOJA, POMME DE TERRE & EMMENTAL	674	8.1	7.2	13.0
3584	GALETTES DE SOJA RECETTE MEDITERRANEENNE	678	7.6	8.6	13.0
4040	BOULE DE SOJA ENROBEE CARAMEL CACAHUETE	678	6.8	10.0	15.0
3619	BURGERS BLE, SOJA	674	5.9	8.5	16.0
3022	STEAK DE SOJA TOMATE MOZZARELLA	674	6.5	6.5	16.7
3583	GALETTES DE SOJA AUX 5 LEGUMES	682	8.0	7.6	13.0
3020	STEAKS DE SOJA A LA PROVENÇALE	678	8.5	3.5	15.5
3025	STEAKS DE SOJA A LA MEDITERRANEENNE	678	8.5	3.5	15.5
6757	CHARCUTERIE A BASE DE SOJA ET POIS	682	11.0	6.3	8.4
2500	HACHE VEGETAL (SOJA OIGNONS ET PERSIL)	674	4.5	13.0	16.0

6. Consideraciones finales

Los resultados presentados son parte del proceso iterativo en el cual se encuentra este Trabajo Final de Máster, todavía se requiere continuar depurando el conjunto de datos para obtener mediante esta metodología un resultado más prolijo con clusters mejor definidos. Se pretende ampliar esta técnica para otros productos consumidos por personas que practican el estilo de vida vegano, además de la incorporación de otras variables tangenciales al consumo de estos productos por área geografica, se detectó que la mayoría de estos productos provienen de Francia, España, Estados Unidos, Canada y Alemania. Se procuró implementar todas las sugerencias realizadas en sesiones previas.

Éstos resultados, así como también los datos se encuentran disponibles en el siguiente enlace del repositorio del proyecto:

- Datos: https://github.com/atilianno/tfm_master_ds/tree/main/1%20-%20BackEnd/Datos
- Script en R TFM_E2_Grupo02.Rmd: https://github.com/atilianno/tfm_master_ds/tree/main/1%20-%20BackEnd/R%20Studio