TECANA AMERICAN UNIVERSITY

POSTDOCTORAL PROGRAM IN BIOSTATISTICS

B911 PROGRAMMING AND STATISTICAL SOFTWARE / PROGRAMACIÓN Y SOFTWARE ESTADÍSTICO

Informe 3

CURSANTE:

Oralia Nolasco Jáuregui

“Por la presente juro que soy la única autora de este documento en el cual se integran opiniones personales en base a mi trabajo y fruto de la investigación bibliográfica”

Jalisco México, Marzo de 2020

Versión 1.0

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

Cluster	De su adaptación en español a clúster. Es un análisis de agrupamiento de datos.
Densidad Gráfica	Es el conjunto de puntos en un espacio limitado, suele describir al cúmulo de puntos en un gráfico.
Disimilaridad	Se refiere a la propiedad de la falta de semejanza o parecido entre dos o más cosas.
Frecuencia Relativa	Se dice del número de observaciones en un intervalo entre el número de datos. Es muy utiliza en los gráficos tipo histogramas.
Percepción Gráfica	Es una función que el cerebro interpreta dependiendo de un conjunto de imágenes en instantes de segundos.
Transformada Fourier	Es una transformación matemática para transformar señales en el dominio del tiempo (o espacial) y el dominio de la frecuencia. Es capaz de transformarse en cualquiera de los dominios al otro.

TECANA AMERICAN UNIVERSITY

Postdoctoral Program in Biostatistics

PROGRAMMING AND STATISTICAL SOFTWARE

Autora: Dra. Oralia Nolasco Jáuregui

Marzo, 2020

RESUMEN

Este documento describe la parte medular de un manuscrito científico, y es la representación gráfica de los datos. La presentación de datos estadísticos por medio de gráficos es considerada una tarea importante en el proceso de comunicación de los datos. Esto no lo desconocen los investigadores a diferentes niveles. Usualmente cuando alguien recibe en sus manos un documento con gráficos, la primer mirada se dirige a éstos. A pesar de la reconocida importancia este proceso no siempre se realiza de la mejor manera. Una ventaja de los gráficos es que pueden mostrarnos cosas que de otra forma hubiese sido muy difícil o imposible. Esta es una de las razones por las cuales casi todo análisis estadístico comienza con gráficos. Una buena descripción informa y obliga al que produce el gráfico a pensar por qué y cómo está presentado el gráfico. Un conjunto de datos menores a 20, no deben presentarse en gráfico sino en una tabla. La distribución de este documento permite ir de menos a más, es decir, en la primera sección de este documento (CAPÍTULO I) describe las generalidades de percepción de un gráfico, el área que un individuo

visualmente puede percibir y la restante se convierte en evitada o datos aburrido. Se dedica el CAPÍTULO II a describir los bloques parte del diagrama que son mínimamente necesarios para lograr una interpretación gráfica correcta de los datos con las herramientas que se encuentran en el software R (ver Ilustración 1). El diagrama de la metodología de la Ilustración 1 es una propuesta del autor de este estudio. Siguiendo el método seleccionado por el autor, se realizó una revisión de la bibliografía y los antecedentes de estudios relacionados con el tema con la finalidad de establecer el conjunto de significados, teorías y conceptos que sirvieran de soporte a la investigación. Se utilizó la metodología cualitativo – interpretativa dada la naturaleza de la investigación. Además que se ejemplifican la extracción de los datos en R con un conjunto de datos histológicos propios del autor, se utiliza una selección de funciones directas en R dependientes del tipo de variable (Univariante o Multivariante) y analiza cada caso en particular para una delicada selección del método gráfico a emplear. En el CAPÍTULO III se profundizan los conceptos de gráficos univariantes y sus implicaciones. Una vez inundado el lector de la información bibliográfica de los gráficos univariantes del capítulo previo, viene a bien describir los gráficos multivariantes en el CAPITULO IV; en este punto el lector será capaz de entender los elementos esenciales de los tipos de gráficos y sus implicaciones de las variables. Finalmente en el CAPÍTULO V de conclusiones, de la investigación bibliográfica y los análisis de los gráficos llevado a cabo a lo largo de la investigación con impresiones de pantalla a manera de ejemplos para demostrar al lector la eficacia de los métodos y del leguaje R. Se integran los objetivos planteados en el CAPITULO V de conclusiones y se sintetizan los conceptos más importantes tratados en el documento; asimismo, se añaden opiniones personales y comentarios que se consideran notables a manera de clausur

INTRODUCCIÓN

Una gran variedad de gráficos de uso corriente en la disciplina estadística y que han probado ser efectivos en la representación de datos (Correa, J., 2002, Pág. 6).

William Playfair es considerado el pionero de la estadística gráfica. Su trabajo en gráficas lo realizó durante más de 36 años. El actuó basado en los siguientes principios que él mismo estableció (Costigan-Eaves y Macdonald-Ross, 1990):

1. El método gráfico es una forma de simplificar lo tedioso y lo complejo.
2. Los hombres ocupados necesitan alguna clase de ayuda visual.
3. Un gráfico es más accesible que una tabla.
4. El método gráfico es concordante con los ojos.
5. El método gráfico ayuda al cerebro, ya que permite entender y memorizar mejor.

Wainer señala que entre la gente es muy común pensar que si un gráfico es bueno, éste deberá ser totalmente comprensible sin ninguna ayuda adicional. Este pensamiento es limitante. Los gráficos "buenos" se dividen en dos categorías (Wainer, H., & Mislevy, R. J., 1990):

1. Un gráfico *fuertemente bueno* muestra todo lo que se quiere conocer sólo con mirarlo.
2. Un gráfico *débilmente bueno*, muestra un gráfico lo que se necesita conocer observándolo, una vez que se explique previamente cómo mirarlo.

Una buena descripción puede transformar un gráfico débilmente bueno en uno fuertemente bueno. Se debe siempre buscar esta trasformación cuando sea posible.

OBJETIVO GENERAL

Dedicar una investigación que escave y profundice en las generalidades y particularidades de los gráficos estadísticos, a manera de extracto de la bibliografía especializada y ofrecer al lector un sumario que propicie la implementación de los gráficos estadísticos con la culta y delicada aplicación e interpretación de los mismos.

OBJETIVOS ESPECÍFICOS

1. Averiguar y plasmar investigaciones exploratorias en bibliografías acerca de las funciones directas y existentes en el lenguaje R.
2. Escudriñar las funciones directas y localizar la formulación de cada función o modelo, ya que es dependiente de un conjunto de específicos y dependencias.
3. Ubicar e instalar el paquete o librería adecuado al gráfico.
4. Calzar los puntos del planteamiento y formulación de las funciones directas para su correcta implementación de acuerdo a las particulares de los argumentos que ellas emplean.
5. Adquirir y transformar los datos de los archivos de extracción, a manera de ejemplificación.
6. Aplicar y emplear el gráfico de acuerdo al tipo de variables involucradas en el conjunto de datos.

CAPÍTULO I

GRÁFICOS ESTADÍSTICOS

No es extraño ver reportes en los cuales se incluyan gráficos donde sólo se presentan dos números, por ejemplo, porcentaje de hombres vs. Porcentaje de mujeres, esto a veces en un gráfico de pie. Si el conjunto de datos es menor a 20, estos deben presentarse en una tabla, ya que usualmente la tabla supera a la gráfica.

Algunos autores discuten que la apariencia es importante y que algunas recomendaciones que son aceptadas, como no conectar con líneas, puntos, etc., pueden ser pasado por alto en algunos casos (Carr y Sun, 1999).

Tufte presenta un índice para medir la cantidad de información irrelevante en un gráfico (Tufte, E. R., Goeler, N. H., & Benson, R., 1990):

$$\text{Razón Datos} - \text{Tinta} = \frac{\text{Tinta de los datos}}{\text{Total de tinta del gráfico}}$$

Este índice también se puede interpretar como el porcentaje de tinta del gráfico que no puede ser borrado sin afectar los datos.

Wainer en una crítica al índice de Tufte, propugna por no sólo la eficiencia en la transmisión de los datos, sino también por la elegancia (Wainer, H., & Mislevy, R. J., 1990).

I.2 Percepción Gráfica

Cleveland and McGill identificaron un conjunto de "tareas elementales de percepción" que las personas ejecutan cuando extraen información cuantitativa de los gráficos. Ellos las

clasifican de acuerdo a la calidad de la percepción como (Cleveland, W. S., & McGill, R., 1986):

1. Posición a lo largo de una escala común.
2. Posición a lo largo de una escala que no está alineada.
3. Longitud.
4. Pendiente (ángulo)
5. Área.
6. Volumen, densidad y saturación de color.
7. Escala de color.

Algunos de ellos pueden parecer obvios, pero otros no.

I.3 Propósito de un Gráfico

¿Para qué es un gráfico? Existen razones que el analista debe esgrimir para justificar la realización de un gráfico. Fienberg refiere a una clasificación de los motivos para presentar un gráfico. Hay gráfico de propaganda, analíticos, sustitutos de tablas, para decoración, entre otros (Fienberg, S. E., & Mason, W. M., 1979).

I.4 Recomendaciones Generales de un Gráfico

Tufte afirma que la excelencia en los gráficos estadísticos consiste de la comunicación de complejas ideas con claridad, precisión y eficiencia. Un gráfico debe (Tufte, E. R., 1983):

- Evadir lo superfluo.
- Utilizar elementos prominentes para demostrar los datos.
- Manejar un par de líneas por cada variable. Hacer que el interior del rectángulo formado por las líneas de escala sean la región de los datos. Colocar marcas afuera de la región de los datos.

- No apeñuscar la región de datos.
- No exagerar el número de marcas.
- Recurrir a una línea de referencia cuando haya un valor importante que deba verse a través de todo el gráfico, pero no dejar que interfiera con los datos.
- No permitir que las marquillas o referencias en la región de datos interfieran con los datos cuantitativos o que se amontonen.
- Evitar colocar notas, marcas o señales a un lado de la región de datos. Colocar notas en el texto o la explicación.
- Gráficos sobrepuestos deben ser visualmente distinguibles.
- La claridad visual debe conservarse bajo reducción y reproducción del gráfico.

I.5 Gráficos Distorsionados

Tufte dice "un gráfico no distorsiona si la presentación visual de los datos es consistente con la presentación numérica". Experimentos sobre la percepción de áreas de círculos sugiere que en el común de las personas al área real siguiendo la fórmula:

$$AP = AR^{\alpha}$$

Donde AP es el Área Percibida, AR es el Área Real y α varía entre 0.5 y 1.1.

Un problema con los gráficos es la intrusión de diseñadores artísticos que piensan que una gráfica es una obra de arte o una caricatura, cuyo fin es advertir al lector y no un medio de comunicación efectivo de los datos. Esas personas tienden a llenar los gráficos de elementos decorativos e inútiles que distraen, ya que piensan que los datos son aburridos. Esto ocurre en los periódicos y revistas populares (Tufte, E. R., 1983).

I.6 Pecados comunes en Gráficos Técnicos

Los gráficos técnicos usualmente no presentan problemas de exceso de decoración, peor a menudo presentan problemas como los siguientes:

1. Falta de título, marquillas y apuntadores.
2. Falta de escala en ejes.
3. Congestionamiento.
4. Escasez de datos.
5. Mala calidad de impresión.

CAPÍTULO II

METODOLOGÍA

La metodología del presente escrito se divide en varios bloques (ver Ilustración 1) y es una propuesta del autor. En primera instancia se tiene la extracción de los datos, desde un archivo tipo Excel (.xlsx o .xls) en R. Estos datos son fueron implantados para ejemplificar el funcionamiento de los gráficos y sus propiedades de manera adecuada, para exponer la correcta representación de los datos. Como se sugiere en el CAPITULO I, es muy importante que la visualización de los datos sea amable y afable.

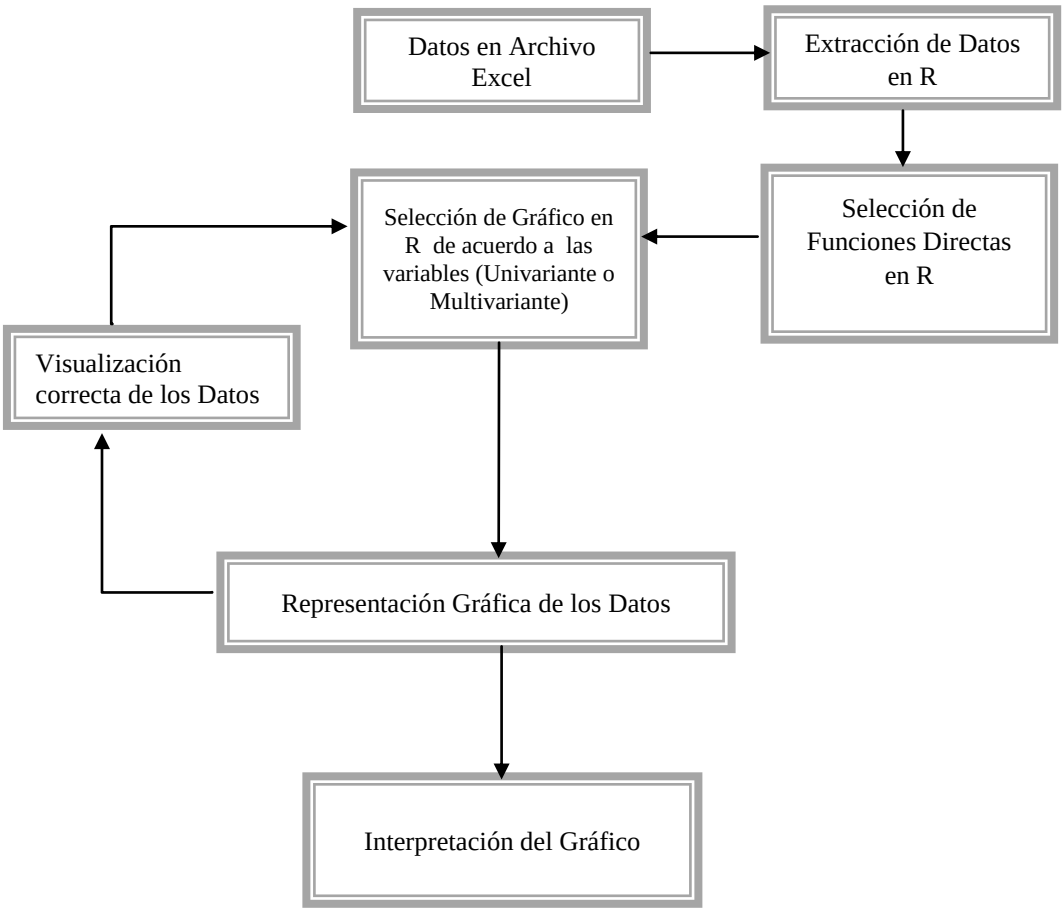


Ilustración 1. Adquisición y tratamiento de datos en R.

II.1 Extracción de datos

Microsoft Excel es una de las aplicaciones más utilizadas para analizar datos y elaborar presentaciones gráficas. Por ello es bastante habitual encontrarse con la necesidad de importar a R información alojada en una hoja de cálculo Excel. Para esta tarea podemos recurrir a múltiples paquetes disponibles en CRAN (Comprehensive R Archive Network) de sus siglas en inglés.

Hay diferentes paquetes que son capaces de extraer los archivos con formato .xls y xlsx, entre ellos se encuentra *XLConnect* y *tidyxl*.

Ambos tienen dependencias de otras librerías, por ejemplo, el paquete *XLConnect* tiene dependencias con *XLConnectJars*(). Asu vez el paquete *tidyxl* tiene dependencia de la librería *tidyverse*(); es importante comentar que ambos tienen necesidad y por ende filiación del paquete *rJava*() y *readxl*().

II.1.1 Paquete XL Connect

Este paquete cuenta con varias funciones capaces de leer el contenido de un archivo Excel y generar un data frame a partir de todo o parte del contenido de una de sus hojas. Una de esas funciones es *readWorksheetFromFile()*.

Abre el archivo Excel indicado por el primer parámetro, recupera el contenido de la hoja indicada por *sheet* y genera un data frame con esa información. Opcionalmente puede indicarse si los datos cuentan con una cabecera o no, mediante el parámetro *header*. Asimismo, puede facilitarse un rango de celdillas a leer mediante el parámetro *región*. Éste contendrá una cadena con un rango tipo “B12:D18”:

readWorksheet (object , sheet , startRow , startCol , endRow , endCol ,

header = TRUE)

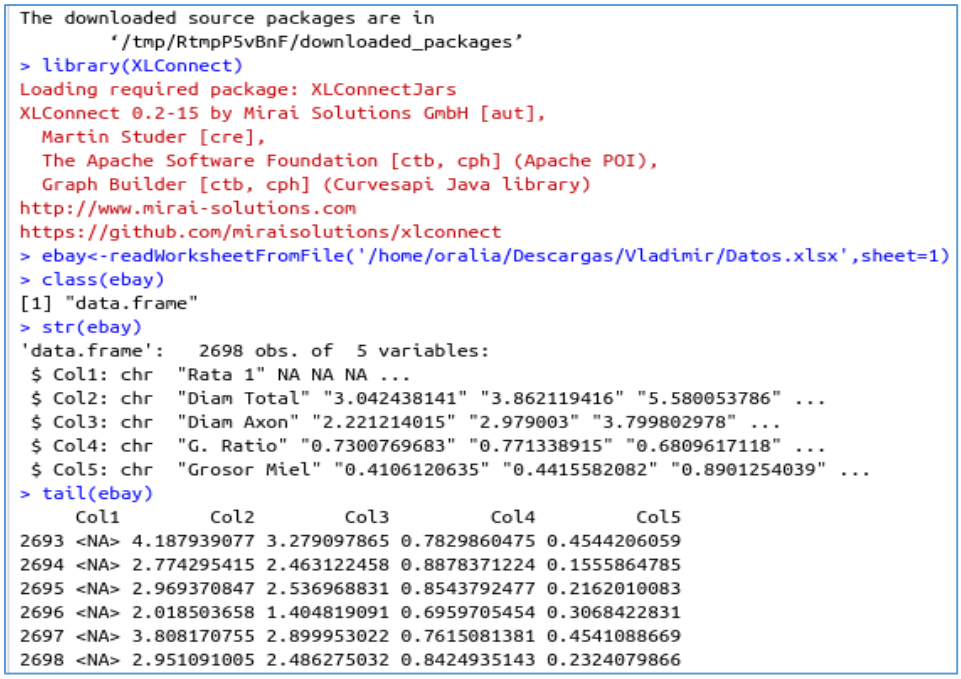


Ilustración 2. Captura de pantalla de la extracción de datos del archivo Histo.xlsx con RStudio en Ubuntu.

II.2 Consola R Herramienta Interactiva

La consola de R es la herramienta básica para operar con R de manera interactiva. Es similar a una línea de comandos del sistema operativo, pero el intérprete procesa sentencias en R en lugar de en Bash o PowerShell (Ojeda, F. C., 2014).

A pesar que desde la consola de R se puede realizar cualquier tarea de forma interactiva, siempre que conozcamos los comandos adecuados y sus parámetros algunas operaciones como la localización y almacenamiento de archivos resultan más cómodas cuando se cuenta con una GUI (Graphics User Interface), aunque se muy básica. Una alternativa, también relativamente sencilla, consiste en integrar R con editores como Emacs o Vim.

II.3 Convertir datos a numéricos

Hay funciones que se consideran imprescindibles en el manejo de datos y para R lo son as.*vector*(), data.*frame*(), as.*numeric*(), entre otras. La instrucción as.*vector*() se necesita para tomar al conjunto de datos como matrices de una dimensión, tomando las características de un vector, se pueden llamar por posiciones (*nrow*(), *dim*()) con un índice, ordenar (de mayor a menor, de menor a mayor), incluso se pueden convertir a una tabla con la función *table*().

La función *data*.frame() se puede considerar de las más utilizadas para trabajar a los elementos de un conjunto como una lista y los elementos internos como objetos.

La función as.*numeric*() se utiliza para trabajar a los elementos como números, también pueden ser del tipo lógico, caracter, complejo, string, double, indefinido (NA) etc.

II.4 Tratamiento de datos NA

Algunas funciones de R están preparadas para trabajar en presencia de datos ausentes (NA), aceptando un parámetro que determina cómo han de ser tratados. Un par de ejemplos de este caso son las funciones *mean*() y *lm*(). La primera acepta el parámetro *na.rm*, con el que se indica si los valores ausentes deben de ser ignorados durante el cálculo o no. La segunda tiene un parámetro llamado *na.action* que, acepta el valor *omit*, con exactamente el mismo resultado: *valores[is.na(valores)]*, *mean(valores)*.

Otra técnica es que en lugar de eliminar posiciones u omitir los elementos del vector o matriz, se pueden sustituir los valores ausentes por un valor a conveniencia.

II.5 Funciones Directas en R

Las funciones permiten realizar las diferentes acciones. Existen funciones definidas (pueden ser modificadas), pero lo más importante es que R permite crear objetos del modo *function*, es decir, accede construir nuevas funciones de R que realicen tareas que no estaban definidas en el momento de instalar el programa, además se puede utilizar a su vez en expresiones posteriores ganando considerablemente en potencia, comodidad y elegancia. Estas nuevas funciones se incorporan en el lenguaje. Para obtener ayuda sobre qué hace una función se utiliza *help*.

Las funciones tradicionales en cualquier lenguaje de uso matemático y estadístico, están definidos en R. Entre las que se encuentran abs, sqrt, sin, cos, tan, asin, acos, atan, exp, log, log10, min, max, sum, prod, length, mean, range, median, var, cov, summary, sort, rev, order.

Una función se define asignando aun objeto la palabre "function" seguida d elos argumentos que se desee asignar, escritos entre paréntesis y separados por comas, seguidos de la orde, entre llaves si son varias órdenes, que desee conceder:

Nombre <- function (arg1, arg2, ..)expresión

La expresión en una fórmula o grupo de fórmulas que utilizan los argumentos para calcular su valor. El valor de dicha expresión es el valor que proporciona R en su salida y éste puede ser un simple número, un vector, una gráfica, una lista o un mensaje.

II.5.1 Media, Varianza, Desviación Típica, Covarianza, Factorial, Progresión aritmética y Progresión Geométrica.

Media: Es una función de un solo argumento (si no se especifica se toman los datos por NA (valor nulo) y el resultado será NA (García, J. M. C., Portillo, E. M., & Cezón, P. A., 2010).

En caso de que se le suministre un argumento, no se comprueba si es válido o no, sin que suponiendo que es un vector, se elimina del mismo todos los elementos correspondientes a NA. Para ello se utiliza la negación (!) y la condición de ser un valor no disponible, ver Figura 2.

Aunque R ya tiene implementada la orden *mean* que calcula la media de los valores que se le indiquen.

Varianza: Hay varias opciones para el cálculo de la varianza:

1. La definición de varianza poblacional, ver Figura 3.

$$Varianza = \frac{\sum(x - \bar{x})^2}{n}$$

2. Se debe de tener en cuenta que T tiene implementada la función *var*, que calcula la varianza muestral o cuasivarianza, por lo que divide por n-1 en vez de n, ver Figura 3.
3. Pero si lo que se quiere es trabajar con var, lo mejor es multiplicar por (n-1)/n.

Desviación Estándar: La definición típica poblacional es $DT = \sqrt{\frac{\sum(x-\bar{x})^2}{n}}$, ver Figura 4.

Función Covarianza: Si se quiere la covarianza muestral se tiene que dividir por n-1 en lugar de n, ver Figura 5.

Logaritmo: Para describir una función que adquiera el logaritmo de varios elementos es importante utilizar elementos de comparación, conjunción, negación, etc. En la Figura 6, se describe el comportamiento de obtención de elementos inversos, y de algunos logaritmos de los elementos de todo el vector o de elementos únicos como el símbolo e, utilizado como el número irracional.

Switch: Tiene una estructura como se describe en la Figura 7.

switch (expresión, [valor-1 =] acción -1, …, [valor-n =] acción –n)

For: Se conforma de una estructura como se describe en la Figura 8.

Factorial: Dos procedimientos alternativos a la función que se muestra en la Figura 9, serían los de la función factorial. Pero la función factorial.3 consume recursos debido a la recursividad, una opción podría ser generar todos los factores antes de multiplicarlos, pero esto consume memoria. Ambas opciones son viables y son elección del programador.

Progresión Aritmética: La progresión aritmética se puede implementar de distintas maneras, las tres más comunes son: la forma explícita, la forma recursiva y la tercera la forma recursiva vectorial. En la Figura 10 se describen la recursiva y la recursiva vectorial.

Progresión Geométrica: Su obtención se verifica a bien en la Figura 11 donde se pueden observar los pasos a seguir para la proceso de la progresión geométrica en R. En la Figura 12, se hace hincapié que es otra manera de obtener la progresión geométrica descrita en la Figura 11.

Detección de Paridad: La identificación de características de los elementos de un vector es fundamental, por ello, se ejemplifica esta función creada para separar elementos pares de los impares, ver Figura 13.

CAPÍTULO III

GRÁFICOS UNIVARIABLES CON R

Uno de los mecanismos de exploración de datos más usuales y útiles consiste en generar representaciones gráficas de las variables que componen el dataset. Es frecuente que a partir de la observación de dichas representaciones pueda obtenerse información más fácilmente interpretable que la nos ofrecen los métodos de exploración descritos en el Capítulo II de metodología.

R cuenta en su paquete base con múltiples funciones para la producción de gráficas, pudiendo generar representaciones en forma de nubes de puntos, líneas, barras, gráficos circulares, etc. También R tiene funciones para elaborar histogramas y curvas de densidad. Los gráficos mencionados con útiles como vía de exploración de los datos, pueden ser almacenados para su posterior reutilización en cualquier tipo de documentación.

El paquete de *graphics*, forma parte de la instalación base de R, por lo que no se necesita una instalación previa ni recargarlo. Se puede obtener una lista de las funciones instaladas y además conocer las funciones del paquete por medio de *library (help = "graphics").*

Los gráficos disponibles en R son todos de gran calidad y poseen una versatilidad muy grande. Para poder demostrar lo anterior a modo de ejemplo, ver la Figura 14, en ella se muestra el resultado de la función *demo ("graphics")*.

II.1 Gráficos Básicos en R

Muchos de los gráficos básicos que es posible generar con R son resultado de la función *plot().* Esta función acepta un importante número de parámetros con los que es posible

configurar el gráfico resultante, establecer títulos y otros parámetros gráficos como colores, tipos de marcadores, grosor de líneas, etc.

plot (valores [, valores Y type = tipográfico, main = tituloPrincipal, sub = subtítulo, xlab = tituloEjeX, ylab = tituloEjeY]

Genera una representación gráfica de los valores facilitado acorde a la configuración especificada por los parámetros adicionales. El parámetro *type* toma por defecto el valor *p*, dibujando un punto por cada dato. Otros posibles valores son *1*, para dibujar líneas y *h*, para obtener un histograma.

Son bastos los tipos de gráficas que se pueden generar con la función plot(), es importante mencionar que también está la versión 3D para plot().

La Tabla I resume los tipos de gráficos y características que pueden ser trabajadas en un diseño gráfico. Además hay funciones como *lines, points segments, arrows* que pueden añadir un plus a los diseños.

Plot() es una función genérica que crea un gráfico en el dispositivo gráfico actual. Igualmente existen funciones específicas en las que funciona de modo especial, como por ejemplo para *data.frame, lm*, entre otras. La Figura 15 ejemplifica un uso básico de *plot*.

En la Figura 15 se ejemplifica la necesidad de conocer el valor máximo (utiliza la función *max()*) y mínimo (con la función *min()*) de los datos a graficar. Con ello se maneja un límite para *xlim()* y también *ylim()*.

Es muy útil el uso de líneas tipo *abline()* que separen las filas en la gráfica, el uso del parámetro *h* es para líneas horizontales y *v* para líneas verticales o columnas, ver Figura 16.

II.1.1 Gráfica de Puntos en R

En la Figura 16 se puede ver el uso de puntos en el gráfico, los puntos rojos son creados de forma aleatorio con una dimensión definida y los puntos azul fueron modificados a un tamaño más grande.

II.1.2 Configuración de títulos y leyendas en R

El uso de nombres en los ejes x, y, z, son imprescindibles para la decodificación correcta de un gráfico, es por ello que la función *plot()* tiene los parámetros *xlab* y *ylab*. Se debe introducir entrecomillada la leyenda que se desea mostrar en los ejes. Con el comando *tex()* se puede introducir algunas leyendas en texto sobre el área del gráfico, de hecho se deben introducir las coordenadas para ubicar el texto.

En la Figura 17 se puede observar el uso de leyendas, nombrado de los ejes y la ubicación de un texto correspondiente a una recta.

La Figura 18 es a muestra de ejemplo con un conjunto de datos histológicos que se han venido tratando a lo largo del documento y que tienen un comportamiento tipo lineal, si sobre ellos se quiere destacar tal comportamiento es necesario ubicar una recta con la función *abline()* y un texto en una posición que indique que pertenece a la recta.

II.2 Segmentos y Flechas en R

La Figura 19 evidencia el uso de los segmentos entre los puntos elementos del vector datos. En ciertas ocasiones es muy importante la búsqueda del trazo del comportamiento entre los elementos y la función *segments()* es ideal para ello.

En la Figura 20 se tiene una flecha del sentido entre los segmentos consecutivos, y para mostrar la dirección se puede utilizar la función *arrows().*

La unión de segmentos dentro del conjunto de muestras puede ser unido en diferentes secuencias, la Figura 21 tiene la asociación de segmentos pares e impares dentro de una población de puntos que se encuentran en un gráfico, y se iluminaron con el color verde.

II.3 Gráfico Pie en R

Los gráficos de pastel son de los más conocidos y utilizados en la industria para representar porcentajes de ventas y/o de la propia economía des las empresas. La fácil interpretación del ser humano y la poca necesidad de un intérprete para ello se ha hecho su diversificación por el mundo sin necesidad de saber incluso del idioma del país donde se presente. Es muy importante para el software de R que esté instalado el paquete de *plotrix* para el uso de los gráficos *pie()*, sobre todo para los tipo *pie3D().*En la Figura 22 se ilustra el uso de un gráfico de elecciones y los porcentajes de los votantes en un prototipo ilusorio para esta captura de imagen.

II.3 Histogramas en R

Los histogramas para variables continuas se representan con la función *hist().* Como se puede ver en la Figura 23, un histograma con datos histológicos propios.

Los datos histológicos son un conjunto de datos normales, el histograma se asemeja a la curva normal. Pero en R asigna el número de intervalos y la amplitud de ellos. Si se quieren modificar, se utiliza el argumento *break*, que indican los puntos de corte junto con los argumentos para darles efectos visuales, ver Figura 24.

Se puede agregar al diagrama histológico una curva sobre la normal para percibir la forma de la distribución. En la Ilustración 3 se añade la curva de color rojo con la función *curve()* que requiere de *rnorm()* para poder dibujar la forma normal.

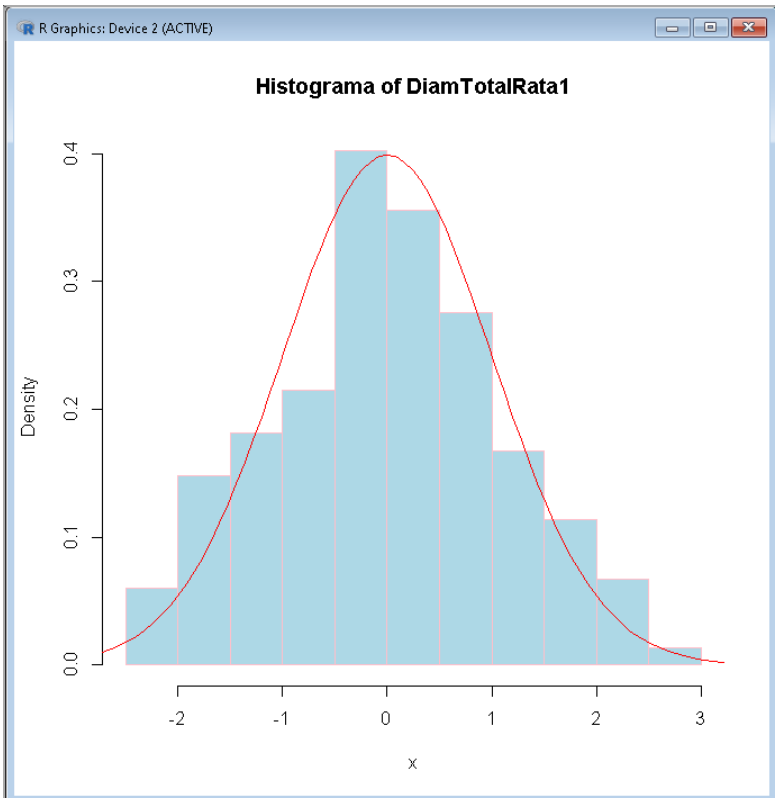


Ilustración 3. Captura de pantalla del uso de hist() con cura de distribución con datos histológicos.

II.4 Gráfica de cajas

Con la función *boxplots()* se pueden crear diagramas de cajas para una o varias funciones. En la Figura 25 describe el proceso de graficado en cajas de un conjunto de datos *mtcars* que son un conjunto de datos ejemplos de R.

Es posible cambiar características como colores y propiedades individuales por cada caja dentro del gráfico, observar Figura 26.

II.5 Gráfica de puntos por escalas en R

Con la función de dotchart() se crean diagramas de puntos por escalas. Se utilizan los datos de R VADeaths, ver Figura 27.

Se pueden transponer los datos y con ello se acota la gráfica. Una variante de este tipo de gráficos es proporcionada cambiando los argumentos de *dotchart*(), ver Tabla II.

II.6 Argumentos modificable en los gráficos de R

II.6.1 *par*()

Entre los argumentos de la función *par*() que permiten representar gráficos múltiples. Así el argumento *mfcol=c(m,n),* divide el gráfico mxn partes iguales, que se rellenan por columnas, análogamente *mfrw=c(m,n)* rellena por filas, ver Figura 28.

II.7 *pairs*()

Esta función permite crear una matriz de diagramas de puntos entre más de dos variables en un solo gráfico, ver Figura 29.

II.8 *scatterplot*()

El uso de los diagramas de puntos *scatterplot*() y en 3D la función *scatterplot3d*() necesitan que se instale la librería del paquete correspondiente. Para indicar los datos que serían parte del gráfico tipo *scatteplot*() necesita la función predecesora *attach*() indicando el conjunto de datos a graficar, ver Figura 30.

II.9 Fractal con R

Como se puede ver en la Tabla I, el entorno gráfico en R es suficiente para desarrollar simulaciones de valores suficientes para construir entornos fractales. En la Figura 31 se visualiza el código de la función *f1*() que es forzoso para una imagen fractal en R;

continuando con una aplicación de las funciones que logran las figuras de fractalidad, la Ilustración 4 tiene el proceso de la función f1() pero con base a los datos histológicos de la Rata 1.

Hay diferentes formas que son tíficas con una esencia fractal, la Figura 32 describe una forma de ellas, que se nombró f2() donde se incluyen argumentos como color, forma y número de puntos; la peculiaridad de esta forma es que es un conjunto de puntos en forma triangular, que se dispersan en base a los tres puntos definidos en la función points() que se auxilian de la función rnorm().

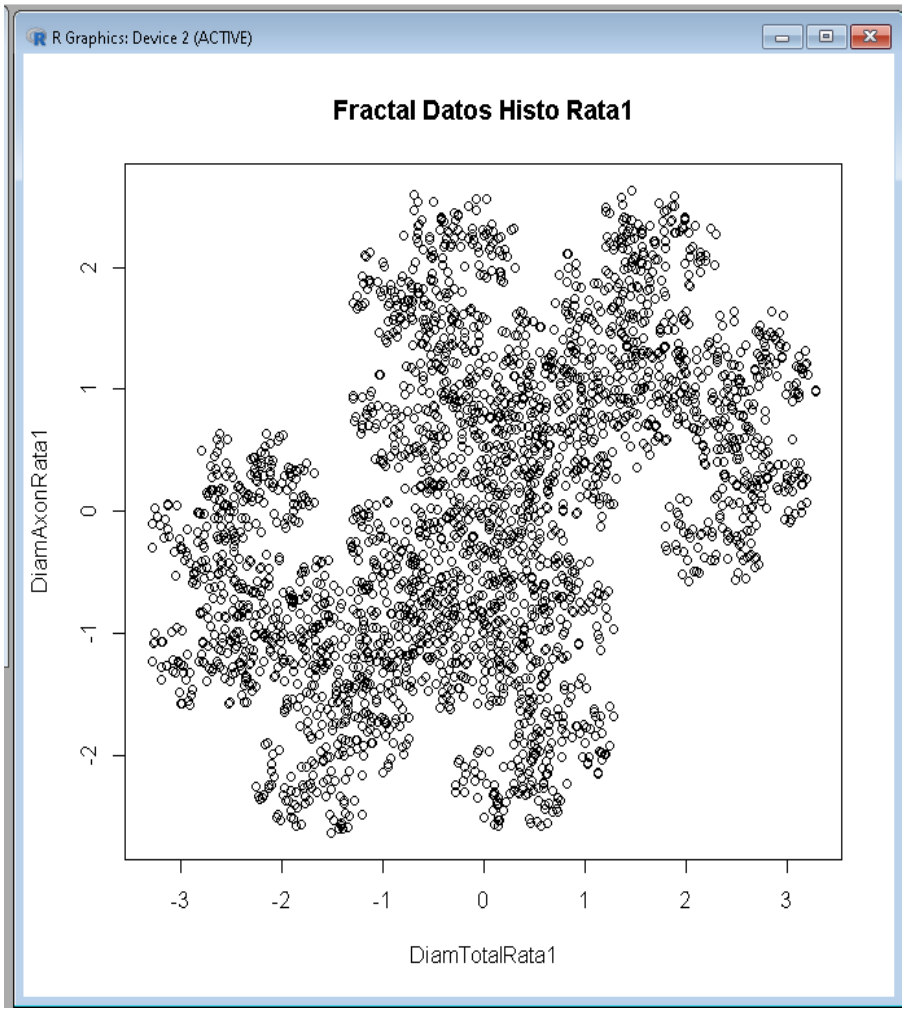


Ilustración 4. Captura de pantalla del uso de fractalidad con datos histológicos.

CAPÍTULO IV

GRÁFICOS MULTIVARIABLES CON R

El análisis Multivariante está relacionado con conjuntos de datos que involucran más de una variable para cada unidad de observación. En general tales conjuntos de datos vienen representados por matrices de datos X con n filas y p columnas, donde las filas representan los casos, y las columnas las variables. En cuanto al análisis de tales matrices, unas veces el estudio se centra los casos, y la mayoría de los casos de estudio en simultáneamente ambas cosas.

III.1 Gráficos de Independencia en R

El gráfico chi-plot constituye n método gráfico asociado a una prueba *no-paramétrica* para probar independencia entre un par de variables continuas, desarrollando por Fisher y Switzer en 2001, el cual complementa la información derivada de un *scatterplot,* en el cual puede resultar difícil juzgar la presencia de algún patrón diferente a una relación monótona.

Estos gráficos pueden ser extendidos al campo de los modelos de regresión para las pruebas de ajuste mediante el análisis de residuales, y para probar si un conjunto de observaciones rezagadas k periodos son incorrelacionadas.

El gráfico se construye a partir de lo siguiente: Sean (x_1, y_1), …., (x_n, y_n) una muestra aleatoria de una función de distribución conjunta (continua) H del par de variables (X,Y), y sea I(A) una función indicadora del evento A. Para cada observación (x_i, y_i), ver Ecuación 1, Ecuación 2, Ecuación 3, Ecuación 4, Ecuación 5 y Ecuación 6.

El *scatterplot* de los pares (Λ_i,X_i),| Λ_i | < $4\left\{\frac{1}{n-1}-\frac{1}{2}\right\}^2$. Λ_i indica la posición del punto (x_i, y_i) respecto al centro de los daos. Adicionalmente, se trazan las líneas horizontales en $\chi = \frac{-cp}{n^{0.5}}$ y $\chi = \frac{cp}{n^{0.5}}$, donde cp se selecciona de forma aproximadamente 100p% de los pares (Λ_i,X_i) estén entre las líneas conocidas como líneas de Fisher y Switzer. Para p =0.90, 0.95, y 0.99, cp= 1.54, 1.78 y 2.18 respectivamente, ver Figura 33.

III.2 Curvas de Andrews en R

Un gráfico de Andrews está basado en transformación de Fourier del conjunto de datos multivariable. Básicamente una transformación de Fourier es una representación funcional alternante de senos y cosenos, de cada observación. La transformación se define como:

$$f(t) = \frac{x_1}{2} + x_2 seno(t) + x_3 \cos(t) + x_4 seno(2t) + + x_5 cos(2t) + \cdots$$

Cada variable de cada observación es representada por una componente individual en la suma de la transformada de Fourier. Tradicionalmente, t varía entre –π y π para permitir una adecuada representación de los datos. La magnitud de cada variable de un sujeto particular afecta la frecuencia, la amplitud y la periodicidad de f, dando una representación única para cada sujeto, ver Figura 34.

Embrechts y Herzberg en 1991, presentan variaciones a los gráficos de Andrews que utilizan polinomios de Chebychey y polinomios de Legendre.

III.3 Gráficos de Estrellas en R

En un conjunto de datos multivariados ordenados matricialmente, de manera que las filas corresponden a las observaciones y p columnas, una por cada variable, se tienen dos dimensiones, que pueden construir círculos (uno por cada observación multivariada) de un

radio prefijado, con p rayos igualmente espaciados emanando del centro de cada círculo, ver Figura 35.

Las longitudes de los rayos son proporcionales a los valores de las variables en cada observación. Los extremos de los rayos pueden conectarse con segmentos de líneas rectas para formar una estrella. Con cada observación representada por una estrella, éstas pueden ser agrupadas según similitudes. Es conveniente estandarizar las observaciones, caso en el cual pueden resultar valores negativos según Johson y Wichern en 1998 lo comprobaron.

III.4 Gráficos de Regresión en R

En regresión simple es posible obtener de manera sencilla los gráficos de dispersión, límites de confianza, recta ajustada y de residuales.

La función *matplot*, grafica las columnas de una matriz contra las columnas de otra, tal que la primera columna de x es graficada contra la primera columna de y, la segunda columna de x contra la segunda de y, etc. Sin embargo si una matriz tiene pocas columnas, éstas son recicladas. Los argumentos son: *x,y, type, lty, lwd, pch, col, cex, xlab, ylab, xlim, ylim, add, verbose*. El primer argumento son los vectores o matrices de datos a graficar, es imprescindible que el número de filas coincida. Si uno de los dos objetos (*x* o *y*) falta, se utiliza como *x* un vector de 1:n, con valores internos tipo NA; *type*: un vector para indicar el tipo de gráfico para cada columna de *y*, por defecto es *p*; *lty,twd*: vectores para indicar los tipos y anchos de líneas para cada columna de *y*; *pch*: carácter o vector de caracteres o enteros para definir los "plotting characters", por cada columna; *col*: vector de colores, aunque si no se especifican los colores son usados cíclicamente; *cex*: vector de dimensiones del factor de expansión de caracteres, es usado cíclicamente; *xlab*, *ylab*: Títulos de ejes x e y; *xlim*, *ylim*: Rangos para los eje; *add*: argumento lógico. Si es TRUE, el gráfico es

agregado a un gráfico previo. Se debe garantizar que los rangos d elos ejes en ambos gráficos sean los mismos; *verbose*: Argumento lógico. Si es TRUE, escribe una línea de lo que hace la función.

III.5 Ajustes de Curvas por Regresión No-Parmétrica en R

III.5.1 Ajuste Spline

En R está disponible en el paquete base la interpolación *spline* cúbica, la cual proporciona bien sea una lista de puntos obtenidos por interpolación o una función que hace la interpolación. La función *spline* devuelve una lista de componentes *x* e *y* que dan las coordenadas en donde se da la interpolación y los valores interpolados. La empleo de *splinefun* es en el caso de realizar una interpolación *spline* cúbica de los puntos dados (a menudo es más útil que la propia *spline*).

Notas: Los splines pueden usarse para extrapolación, es decir, para predecir en puntos por fuera del rango de la variable x. Sin embargo si el método usado es *fmm*, dicha extrapolación tiene poco sentido. Si se trata del método "natural", esta extrapolación es lineal usando la pendiente de la curva de interpolación en el punto de datos más cercano. También se puede utilizar los argumentos *approx* y *approxfun* para interpolación constante y lineal. En el paquete *splines*, se tiene *interpSpline* y *periodicSpline*, son generalmente utilizados para generar las bases *spline* que pueden ser usados para regresiones *spline*. En el paquete *modreg* se tiene *smooth*.spline para *splines* con suavizado.

III.5.2 Regresión Kernel en R

En R se tiene el paquete o librería *modred* la función *ksmooth* con la cual es posible obtener la curva ajustada con la función *kernel* determinado, utilizando el estimador *Nadaraya-Watson*. Esta función produce una lista con componentes *x*: los valores en orden creciente,

en los cuales es evaluado el ajuste suavizados; y *y* los valores ajustados correspondientes a *x*. Los *kernels* son escalados de modo que sus cuartiles (vistos como densidades de probabilidad) estén en +/- 0.25**bandwith*. El rango de *x* (x.*range*) son los puntos a ser cubiertos en la salida (*output*) o resultado. El argumento n.*points* es el número de puntos en los cuales se va a evaluar el ajuste y x.*points*, son los puntos en los cuales se evaluará el ajuste suavizado. Si no se especifica entonces n.*points* será elegido uniformemente para cubrir x.*range*.

III.5.3 LOESS en R

El ajuste de polinomios locales propuesto por Cleveland, puede lograse en R mediante la función *loess* del paquete *modreg*. La expresión matemática para *loess* especifica la respuesta y uno o más predictores numéricos. En secuencia, primero busca en los datos la respuesta, después los predictores y enseguida sustituye la respuesta con el valor de los pesos. Los pesos son opcionales en cada caso. *Loess* tiene una especificación opcional en sus argumentos y es el uso de subconjuntos. La acción a seguir con valores faltantes en la respuesta o en los predictores es definitivamente detener la evaluación. El parámetro α es el que controla el grado de suavizado, además tiene un parámetro *degree* para el elevación del polinomio a ser manejar. Para los ajustes de los predictores deben ser del tipo cuadrático (recomendado), los predictores igualmente pueden ser normalizados a una escala común si hay más de uno. La normalización aplicada consiste en aprovechar la desviación estándar recortada al 10% y ajustar el parámetro *normalize* que exige saber que los predictores en coordenadas espaciales y *otros* están en escala común. En el parámetro *family* se especifica como *gaussian* si el ajuste se desea por mínimos cuadrados, y se especifica *symetric* si un

estimador M redescendiente se destina como una función *biponderada* de *Tukey*. El ajuste de *loess* se hace con el paquete *model.frame* y los parámetros de control con *loess.control*.

Notas: El ajuste es ejecuta localmente. Es decir, para el ajuste en un punto *x*, el acomodo es hecho usando los puntos en una vecindad de *x*, ponderado por sus distancias desde *x* (con diferencias en *parametric* las variables son ignoradas cuando se calcula la distancia). El tamaño de la vecidad es controlado por α (establecido por *span* o por *enp.target*). Para $\alpha < 1$, la vecindad incluye la proporción α de los puntos, y estos tienen una ponderación tricúbica (proporcional a $(1 - (dist/maxdist))$. Para $\alpha > 1$, todos los puntos son usados, con la distancia máxima asumida como α / p veces la distancia máxima actual para *p* variables explicativas.

Para la familia por defecto, el ajuste es por mínimos cuadrados (ponderados). Para *symetric* son destinadas unas pocas iteraciones de un procedimiento de estimación M con *biponderación* de *Tukey*. Tener en cuenta que como el valor inicial es el ajuste de mínimos cuadrados, éste no pide ser un ajuste muy reciente. Puede ser importante ajustar la lista de control para alcanzar una velocidad aceptable, con *loess.control* se logra.

Si se quiere obtener un gráfico simultáneo de dispersión con la curva ajustada por *Loess*, la función *scatter.smooth* del paquete *modreg* permite tal gráfico, en tanto que la función *loess.smooth* sólo nos proporciona una lista de valores de *x* e *y*.

Alternativamente, también es posible graficar la curva concertada por Loess junto con el *scatterplot* manejando la función *loess.smoth*, y con la cual se pueden superponer diferentes curvas ajustadas.

III.6 Análisis de Componentes Principales en R

Los gráficos básicos en el análisis de componentes principales son:

1. *Scores-Plot*
2. Scree-Plot
3. *Principal Components Pattern Plot*

En la Figura 36 se presentan 5 eventos u opciones para obtener las componentes principales dada una matriz de datos de dimensiones *n x p*, donde las *n* filas corresponden a las observaciones y las *p* columnas a los valores de las correspondientes *p* variables, así como resultados de dos aplicaciones.

Utilizando las deducciones obtenidas en la Figura 36, con *res <- compprinc2(x)* se procede a graficar el *Scores-Plot* para los datos de la matriz X.

Para los resultados adquiridos estandarizando los datos, se construyen a continuación el *Scree-Plot*, el *Scores-Plot* y el *Principal Components Pattern Plot*, como se ilustran en la Figura 37 y Figura 38.

III.7 Análisis de Agrupamientos (Clusters)

Para ejecutar el análisis *cluster* en R es necesario cargar los siguientes dos paquetes:

- library (mva)
- library (cluster)

No obstante asimismo en el paquete survival existe una función *cluster*. Se enlistan las funciones relacionadas con el análisis de *cluster* disponibles en R: *clara.objet(), clara(),*

clusplot.default(), cusplot(), plot.agnes(), plot.diana, plot.mona(), pltree(), pltree.twins(), *twins.object(), xlcara(), hclust(), identify.hclust(), kmeans(), rect.hclust(), cluster().*

III.7.1 Análisis de la función cusplot, culspolt.default y clusplot.partiti en R

La situación de *clusplot* es una función genérica, que dibuja un *clusplot* de dos dimensiones sobre el dispositivo gráfico actual. Posee un método por defecto (*cluster-plot.default*) y partición (clusplot.partition).

III.7.1.1 Análisis de Clusplot.default

Crea un gráfico bivariado para visualizar una partición (*clustering*) de los datos. Todas las observaciones son representadas por puntos en el gráfico, usando escalamiento por componentes principales o multidimensionales. Una elipse es dibujada alrededor de cada *cluster*. Utiliza una matriz de disimilaridad, según el valor del argumento *diss*. En caso de ser una matriz, cada fila corresponde a una observación, y cada columna corresponde a una variable. Todas las variables deben ser numéricas.

Nota 1: Valores faltantes (NA) son permitidos, los cuales son remplazados por la mediana de la variable correspondiente. Cuando alguna variables o algunas observaciones contienen sólo valores faltantes, la función para cada un mensaje de advertencia.

Nota 2: En caso de ser una matriz de disimilaridad, *x* es el resultado de la función *daisy* ó *dist* ó una matriz simétrica. Además puede ser un vector de longitud n*(n-1)/2, donde n es el número de observaciones, y será interpretado de la misma forma como la salida de las funciones antes mencionadas. Valores faltantes no son permitidos.

El parámetro *clus*, es un vector de longitud *n* que representa un *clustering* de *x*. Para cada observación el vector lista el número o nombre del *cluster* al cual ha sido asignada. A

menudo *clus* es la componente *clustering* de la salida de la función *pam*, *fanny* o *clara*. El parámetro *diss*, es un argumento lógico que indica si x será considerada como una matriz de disimilaridad o una matriz de observaciones por variables. La medida *lines*, es un entero de 0, 1, 2, usado para obtener una idea de las distancias entre elipses. La distancia entre dos elipses es medida a lo largo de la línea que conecta sus centros. Si la elipse se sobreponen en la línea que une a sus centros, ésta no es dibujada. De lo contrario, el resultado depende del valor de este argumento:

- lines = 0: no aparecerá ninguna línea de distancia.
- lines = 1: aparecerá el segmento de línea que une los centros de las eclipses.
- lines = 2: es dibujado un segmento de línea entre los bordes de las elipses.

Las elipses son sombreadas en relación a sus densidades. La densidad es el número de puntos en el *cluster* dividido por el área de la elipse. Incluso las elipses pueden ser coloreadas con respecto a sus densidades, a medida que se incrementa la densidad, los colores son azul claro, verde claro, rojo y púrpura. Cuando se tienen 4 o menos *clusters*, entonces el color = TRUE da a cada *cluster* un color diferente. Cuando hay más de cuatro *clusters*, *clusplot* usa la función *pam* para particionar las densidades en 4 grupos tales que las elipses con la misma densidad más cercana tengan el mismo color. Los valores del vector *clus* son tomados como etiquetas para los *clusters*. Las etiquetas de los puntos son los nombres de las filas de *x* cuando *x* es una matriz. De lo contrario, x es una vector, y las etiquetas de los puntos pueden ser vinculados a x como un atribulo labels, ocmo sucede para la salida de función *daisy*. No deberá tomarse en cuenta un posible atributo *names* de *clus*. Si *span* es TRUE cada *cluster* es representado por la elipse con el área más pequeña que contenga a todos los puntos (este es un caso particular del elipsoide de volúmenes

mínimos. Si es FALSE la elipse está basado en la media y la matriz de covarianza de sus puntos, a menudo produce una elipse mucho mayor. Incluso en algunos casos especiales, cuando un *cluster* consiste de sólo un punto, un círculo diminuto es dibujado alrededor de éste. Cuando los puntos de un *cluster* caen sobre una línea recta con *span*=FALSE, dibuja una elipse angosta alrededor de ésta y si *span*=TRUE, dibuja el segmento de línea exacto.

Nota 3: *clusplot* usa las funciones *princomp* y *cdmscale*, que corresponden a técnicas de reducción de datos y que representarán a los datos en un gráfico bivariado.

Si *lines* es 1 ó 2 se obtiene una matriz cuadrada de orden *k*, donde *k* es el número de *clusters*. El elemento en *[i,j]* es la distancia entre las elipses *i* y *j*. si *lines* = 0, entonces el valor de esta componente es NA.

Un vector de longitud *k* donde *k* es el número de *clusters*, que contiene la cantidad de sombreado por *cluster*. Sea *y* un vector donde el elemento *i* es el número de puntos en el *cluster i* dividido por el área de la elipse *i*. Cuando el *cluster i* es un segmento de línea *y[i]* y la densidad del *cluster* son ajustados a NA. Sea *z* la suma de todos los elementos de *y* sin los NA's, entonces el sombreado es igual a *y/z**37+3.

III.7.1.2 Análisis de la función clusplot.partition en R

Realiza un gráfico bivariado *cluspolot* (*clustering plot*) de un objeto tipo partición. Un argumento de clase *partition*, puede ser creado con las funciones: *pam*, *clara* o *fanny*. En todos los argumentos opcionales disponibles para *clusplot.default* (excepto el correspondiente a *diss*) pueden proporcionarse para esta función. Además, pueden proporcionarse parámetros gráficos como *par*.

Si los algoritmos para generar *clusters pam*, *fanny* y *clara* son aplicados a una matriz de datos de observaciones por variables, entonces un *clusplot* resultante del clustering puede dibujarse siempre. Cuando la matriz de datos contiene valores faltantes y el *clustering* es realizado con *pam* o *fanny*, la matriz de disimilaridad será dada como entrada para clusplot. Cuando se aplica el algoritmo *clara* a la matriz de datos con valores NA's entonces el *clusplot* reemplazará a estos como se explicó en *clusplot.default*, dado que a está disponible una matriz de disimilaridad.

III.7.2 Función hclust en R

Esta función desarrolla un clustering jerárquico sobre un conjunto de disimilaridades. Está a cargo de del análisis de *cluster* usando un conjunto de *n* objetos a ser agrupados. Inicialmente, cada objeto es asignado a su *cluster* y entonces el algoritmo procede iterativamente en cada etapa juntando los dos *clusters* más similares, hasta que haya un solo *cluster*. En cada etapa las distancias entre *clusters* son recalculadas por la fórmula de disimilaridad de Lance-Williams según el método de *clustering* usado.

El método de varianza mínima de Ward busca *clusters* compactos, esféricos. Los métodos de *linkeo complete hallan clusters* similares. El método de *linkeo single* (el cual está cercanamente relacionado al *minimal spanning tree*) adopta la estrategia *clustering* de amigos de amigos. Los dos métodos pueden ser considerados como apuntando al hallazgo de *clusters* con características entre los métodos de *linkeo single* y *complete*.

III.7.3 Series de Tiempos en R

Las series de tiempos pueden ser analizadas en R usando las funciones disponibles en los paquetes bas, boot y ts, con las siguientes funciones (entre paréntesis ubica al paquete):

- plot.ts(base): Grafica objetos que constituyen series de tiempo.
- print.ts(base): Imprime series de tiempo.
- start(base): Extrae y codifica los tiempos de la primera y última observación de las series de tiempos. Se proporciona sólo para compatibilidad con S.
- time(base): Muestra tiempos de series de tiempos.
- ts(base): Crea objetos tipo series de tiempos.
- acf(ts): Estima la función de autocovarianza de autocorrelación.
- pacf(ts): Estima la función de autocorrelación parcial.
- tsp(base): Retorna el atributo tsp de objetos como series de tiempo. Disponible para compatibilidad con S.
- tsboot(boot): Bootstrapping de series de tiempo.
- ar.ols(ts): Ajusta modelos AR a series de tiempo por OLS (mínimos cuadrados ordinarios).
- ar(ts): Ajusta modelos AR a series de tiempo.
- arima0(ts): Modelación ARIMA de series de tiempo. Versión preliminar.
- embed(ts): Encaja una serie de tiempo.
- filter(ts): Filtra una serie lineal de un tiempo.
- lag.gplot(ts): Grafica series de tiempo contra sus versiones rezaadas.
- lag(ts): Rezaga una serie de tiempo.
- diff.ts(ts): Diferencia una serie de tiempo.

CAPÍTULO V

CONCLUSIONES

En este capítulo se presenta a manera de cierre a la investigación bibliográfica de las diversas aplicaciones de las funciones estadísticas comunes y específicas del entorno científico que son mínimamente necesarias para extraer y analizar los conceptos más comunes en un conjunto de datos.

Es muy importante mencionar que se desea explicar al lector que hay muchas variantes de las funciones y que son bastos los paquetes y librerías que se pueden encontrar en la web en la actualidad. En este ejercicio se muestran sólo algunas de los tantos tipos que existen y de analizan con datos propios y propuestos por los autores de los documentos anexados en la bibliografía.

IV.1 Conclusiones

Con el objeto de esquematizar el cuerpo de conclusiones, se agruparán atendiendo a los objetivos.

IV.1.1 Objetivo General

Se brindó una descripción basta que se escatima ser la justa para que cualquier lector del área o ajeno a ella sea capaz de involucrase en la implementación gráfica de los conceptos estadísticos a manera que el escrito lleva de la mano al lector de menos a más; iniciando con el CAPITULO I que surge de la necesidad de invocar el correcto uso gráfico y activar las apropiadas percepciones en el individuo y provocar en el artículo que la parte medular de él sea un buen diseño visual.

De esta manera, se presentó una metodología apegada a las generalidades concurrentes en las bibliografías consultadas para un ajuste e implementación de los argumentos y modelos estadísticos que se encuentran en los paquetes dedicados a la presentación visual de los datos disponibles en CRAN (ver Ilustración 1).

IV.1.2 Objetivos Específicos

En cuanto a estos objetivos, las implementaciones de las funciones directas como en funciones particulares de los autores bibliográficos, permitieron a la autora del trabajo, identificar los aspectos a considerar en el *graficado* de los datos, son dependientes del nivel de información disponible en el manejo de los argumentos.

1. Se averiguó y plasmó a lo largo del escrito, las investigaciones exploratorias en las bibliografías especializadas dedicadas a describir y documentar las funciones directas y se justificó su uso en las figuras anexas en la sección final del documento, donde se hicieron captura de pantalla de los resultados obtenidos en lenguaje R.
2. Se escudriñó el comportamiento de las funciones directas tanto teóricamente (ver sección de Ecuaciones) como en su implementación en el software. Incluso se localizó en la web, dependencias o paquetes dependientes.
3. Se instalaron en el computador, de acuerdo a la documentación técnica de ayuda, el paquete o librería adecuado a cada gráfico que se fue documentando y describiendo a lo largo de la investigación.
4. Se calzaron los puntos indispensables para cada implementación descrita en los CAPITULO III y CAPITULO IV correspondiente a los gráficos univariantes y multivariantes, de acuerdo a las particulares que cada argumento emplea.

5. Se extrajeron, aplicaron y emplearon datos de un archivo de Excel que contiene datos histológicos del sistema periférico de un roedor del laboratorio de la autora, a manera de ejemplificación y demostración de las funciones gráficas.

IV.1.3 Opinión Personal

El manejo de los argumentos es delicado. Es decir, es imprescindible empaparse de la ayuda que ofrecen los tutoriales para cada tipo de modelo y o ecuación involucrada en la formulación estadística, con un solo argumento que no sea adecuadamente definido los resultados no serán los adecuados, por ende el tratamiento no será adecuado y el gráfico no será el correcto. Con ello, es notable que esta investigación fue lenta y llena de trabajo arduo sobre todo en el CAPITULO IV que involucra los gráficos multivariables.

Dentro de las particularidades de los gráficos multivariables es saber detalles como:

- las curvas de Andrews, para ellas es necesario instalar el paquete que lleva su nombre.
- para la gráfica de los componentes principales, score-plot es necesario el paquete pls o plsr y stats.

BIBLIOGRAFÍA

Carr, D., & Sun, R. (1999). *Using layering and perceptual grouping in statistical graphics*. Computing Science and Statistics, Vol. 10, No.1, Pp. 69-73. Center for Computational Statistics George Mason University. dcarr@galaxy.gmu.edu.

Cleveland, W. S., & McGill, R. (1986). *An experiment in graphical perception*. International Journal of Man-Machine Studies, Vol. 25, No.5, Pp. 491-500. Elsevier Editorial.

Correa, J., & González, N. (2002). *Gráficos Estadísticos con R*. Posgrados en Bioestadística de Universidad Nacional de Medellín. Editorial Universidad Nacional de Colombia.

Fienberg, S. E., & Mason, W. M. (1979). *Identification and estimation of age-period-cohort models in the analysis of discrete archival data*. Sociological Methodology, Volumen 10, Pp. 1-67. ASA (American Sociological Association) Editorial. DOI: 10.2307/270764

García, J. M. C., Portillo, E. M., & Cezón, P. A. (2010). *Introducción a la Programación Estadística con R para profesores*. Grupo de Educación Estadística. Editorial Universidad de Granada. ISBN: 978-84-693-4859-8.

Ojeda, F. C. (2014). *Análisis exploratorio y visualización de datos con R*. Departamento de Estadística. Editorial Universidad Carlos III de Madrid (Uc3m). Disponible en: http://www.fcharte.com/libros/ExploraVisualizaConR-Fcharte.pdf

Patricia Costigan-Eaves & Michael Macdonald-Ross. (1990). *William Playfair (1789-1823)*. Statistical Science, Vol. 5, No. 3, Pp. 318-326. JSTOR Editorial. Available on: https://www.jstor.org/stable/i313071

Tufte, E. R. (1983). *The visual display of quantitative information.* Second Edition. Published by Graphics Press, Box # 430, Cheshire, Connecticut, EE.UU.

Tufte, E. R., Goeler, N. H., & Benson, R. (1990). *Envisioning information*. Vol. 126. Published by Graphics Press, Box # 430, Cheshire, Connecticut, EE.UU.

Wainer, H.; Dorans, N.J.; Flaugher, R.; Green, B.F.; Mislevy, R.J.; Steinberg, L.; Thissen, D. (1990). *Computerized Adaptive Testing: A Primer*. Hillsdale, New Jersey. Lawrence Erlbaum Associates Publishers, Pp. 65-102.

Forsythe, G. E., Malcolm, M. A. and Moler, C. B. (1977). *Computer Methods for Mathematical Computations*. Vol. 259, Englewood Cliffs, New Jersey. Prentice-Hall Editorial.

W.S. Cleveland, E. Grosse and W.M Shyu (1992). *Local Regression Models*. Chapter 8 of Statistical Models. Edited by Seds J.M. Chambers and T.J. Hastie. Wadsworth Brooks/Cole Mathematics Series.

Kaufman, L. Rousseeuw, P. J. (1990). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley Series in Probability and Statistics. New York, EE.UU.

Pison, G. Struyf, A. and Rousseeuw, P.J. (1999). *Displaying a clustering with CLUSPLOT*. Computational Statistics and Data Analysis. Vol. 30, No. 4, Pp. 381-392. University of Antwerp. Elsevier Editorial.

Struyf, A., Hubert, M. And Rousseeeuw, P. J. (1997). *Integrating Robust Clustering Techniques in S-PLUS*. Computational Statistics and Data Analysis. Vol. 26, No. 1, Pp. 17-37. Elsevier Editorial.

WEBGRAFÍA

WU (Wirtschaftsuniversität Wien). (2020). *XLConnect Package.* The Comprehensive R Archive Network. 13 de febrero de 2020, de Institute for Statistics and Mathematics. Website: https://cran.r-project.org/web/packages/XLConnect/vignettes/XLConnect.pdf

WU (Wirtschaftsuniversität Wien). (2020). *Tidyxl Package*. The Comprehensive R Archive Network. 13 de febrero de 2020, de Institute for Statistics and Mathematics. Website: https://cran.r-project.org/web/packages/tidyxl/tidyxl.pdf

Marcelo S. Perlin (marcelo.perlin@ufrgs.br). (2020). *Importing Data from Local Files*. 15 febrero 2020. MSPELIN. Website: https://www.msperlin.com/pafdR/importing.html

Stack Overflow. (2020). *Error when using readxl does not exist*. 17 Febrero 2020. Stack Overflow Company. Website: https://stackoverflow.com/questions/39495737/error-when-using-readxl-exdir-does-not-exist

DM. (2020). Get *started in Data Science With R*. 27 Febrero 2020. Data Mentor. Website: https://www.datamentor.io/r-programming/factor/

Departamento de Estadística. (2020). Errores Comunes R. 20 febrero 2020. Universidad Politécnica de Cataluña. Sitio web: http://www-eio.upc.es/teaching/best/R/ErroresComunesR.pdf

ANEXOS

TABLAS

Comando en R	**Proceso**
abline()	Función que añade una o más líneas rectas
plot()	Función genérica para representar en el plano xy puntos, líneas, etc.
barplot()	Diagrama de barras
pie()	Diagrama de sectores
hist()	Histogramas
boxplot()	Diagramas de box-and-whisker
stripplot()	Similares a boxplot() con puntos
sunflowerplot()	Representación en el plano xy de diagramas de girasol
qqnor()	Diagramas de cuantil o cuantil frente a la distribución normal
qqplot()	Diagramas de cuantil o cuantal de dos muestras
qqline()	Representa la línea que pasa por el primer y tercer cuartil.
lines()	Añade líneas a un gráfico
points()	Añade puntos a un gráfico
segments()	Añade segmentos a un gráfico
arrows()	Añade flechas a un gráfico
polygons()	Añade polígonos a un gráfico
rect()	Añade rectángulo a un gráfico

abline()	Añade una recta de pendiente e intersección dada
curve()	Representa una función dada

Tabla I. Funciones Gráficas Básicas en R.

```
x$color[x$cyl==6] <- "blue"
> x$color[x$cyl==8] <- "darkgreen"
> dotchart(x$mpg, labels=row.names(x),
cex=0.7,groups=x$cyl,
+ main="Kilimetraje segun modelo de coche",
+ xlab="Kilometro por litro", gcolor="black", color=x$color)
```

Tabla II. Segmento de código en R para hacer modificaciones en los argumentos de dotchart() y hacer cambios de colores por secciones.

FIGURAS

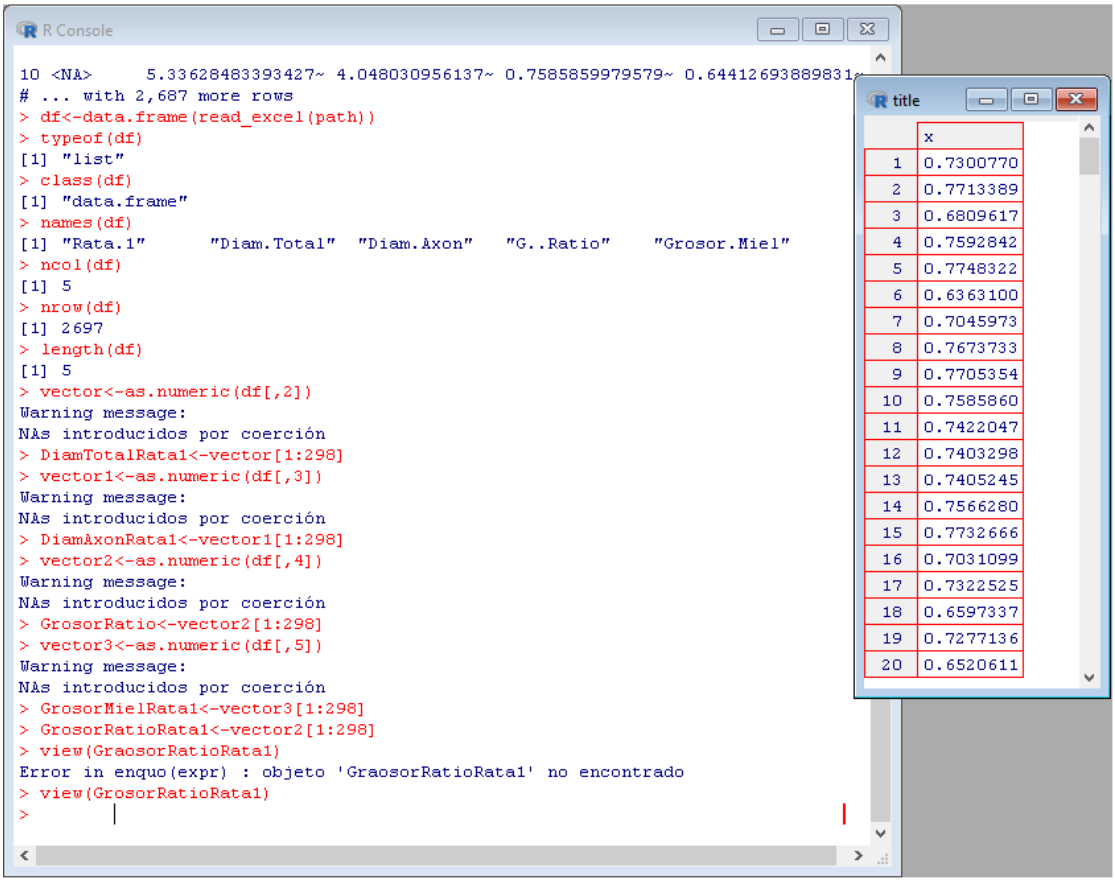


Figura 1. Captura de pantalla de la extracción de datos de un archivo Histo.xlsx con data.frame; extracción, uso y modificación de vectores numéricos en Rstudio en Windows.

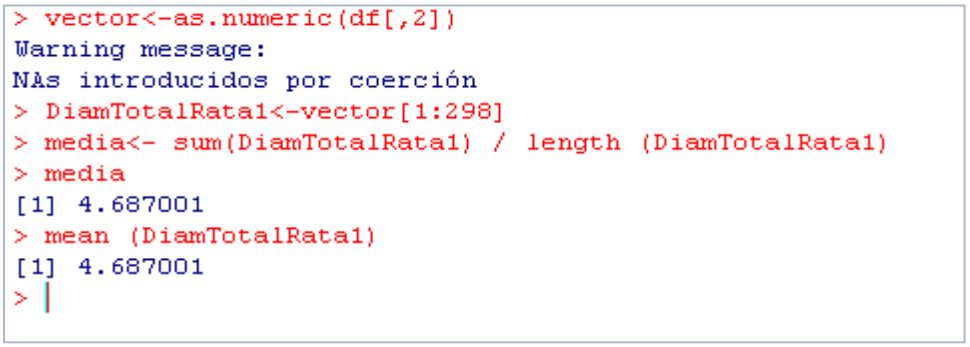


Figura 2. Captura de pantalla del cálculo de función media por dos métodos distintos.

```
> n <- length(DiamTotalRata1)
> v <- sum (( DiamTotalRata1 - (sum (DiamTotalRata1) / n ))^ 2)/ n
> v
[1] 1.239646
> var (DiamTotalRata1)
[1] 1.24382
> # Dividiendo por n-1 debe ser igual a var
> v <- sum (( DiamTotalRata1 - (sum (DiamTotalRata1) / n ))^ 2)/ (n-1)
> v
[1] 1.24382
>
```

Figura 3. Captura de pantalla del cálculo de función varianza y cuasivarianza por dos métodos distintos.

```
> #Desviación Estándar
> n <- length(DiamTotalRata1)
> x <-DiamTotalRata1
> DT <- sqrt ( sum ((x - (sum (x) / n)) ^ 2 ) / n)
> DT
[1] 1.113394
> # Si se quiere la Desviación Estándar  de n y no (n-1)
> DT <- sqrt ( sum ((x - (sum (x) / n)) ^ 2 ) / (n-1))
> DT
[1] 1.115267
>
```

Figura 4. Captura de pantalla del cálculo de función desviación estándar n y n-1.

```
>
>
> #Covarianza poblacional de dos conjuntos de datos
> x<-DiamAxonRata1 <- vector1 [1:201]
> y<-DiamAxonRata2 <- vector1 [302:502]
> length (x)
[1] 201
> length (y)
[1] 201
> n <- length(x)
> cv <- sum (( x- (sum (x)/ n)) ^ 2 * ( y - (sum (y)/n)) ^ 2) / n
> cv
[1] 1.096296
> #covarianza muestral es n-1 en lugar de n
> cv <- sum (( x- (sum (x)/ n)) ^ 2 * ( y - (sum (y)/n)) ^ 2) / (n-1)
> cv
[1] 1.101778
>
```

Figura 5. Captura de pantalla del cálculo de función covarianza entre con n y n-1.

```
> # Funciones del logaritmo
> log (3)
[1] 1.098612
> log (10)
[1] 2.302585
> log (exp(1))
[1] 1
> #Funcion inversa
> Inverso <- function (DiamAxonRata1)
+ ifelse ( DiamAxonRata1 == 0 , NA, 1 / DiamAxonRata1)
> Inverso (-2 : 3)
[1] -0.5000000 -1.0000000         NA  1.0000000  0.5000000  0.3333333
> Inverso (-10 : 10)
 [1] -0.1000000 -0.1111111 -0.1250000 -0.1428571 -0.1666667 -0.2000000 -0.2500000 -0.3333333 -0.5000000 -1.0000000         NA
[12]  1.0000000  0.5000000  0.3333333  0.2500000  0.2000000  0.1666667  0.1428571  0.1250000  0.1111111  0.1000000
```

Figura 6. Captura de pantalla del cálculo de función logaritmo y función Inverso.

```
> #Funcion switch
> n <- 3
> switch (n, "Uno", "Dos", "Tres")
[1] "Tres"
> n <- 1
> switch (n, "Uno", "Dos", "Tres")
[1] "Uno"
```

Figura 7. Captura de pantalla de los elementos necesarios para la función switch.

```
> #función for
> for ( i in -5 : 5)
+ {cat (i, "\t", i^3 , "\n") }
-5      -125
-4      -64
-3      -27
-2      -8
-1      -1
0       0
1       1
2       8
3       27
4       64
5       125
>
```

Figura 8. Captura de pantalla de los elementos necesarios para la función for.

```
> #Funcion Factorial
> factorial.3 <- function (n)
+ { f <-1
+ while (n > 0)
+ ()
+ ()
+ return (f)
+ }
>
>
>
> #Funcion Factorial
> factorial.3 <- function (n)
+ {
+ f <-1
+ if (n > 1)
+ for ( i in 1:n)
+ f <- f * i
+ return (f)
+ }
> factorial (3)
[1] 6
> factorial (25)
[1] 1.551121e+25
> factorial (0)
[1] 1
>
```

Figura 9. Captura de pantalla de los elementos necesarios para la función factorial.

```
>
> #Progresion Aritmetica
> arit.2<-function (n = 1, a1 =1, d =1 )
+ {
+ if (n>1)
+ {return (arit.2 (n-1, a1,d)+d)}
+ else
+ (return(a1))
+ }
> arit.3 <- function (n = 1, a1 =1, d=1)
+ {
+ A=1:n
+ A[1]=a1
+ for (i in 2:n) A[i]=A[i-1]+d
+ return (A[n])
+ }
> arit.2(10)
[1] 10
> arit.3(10)
[1] 10
> #Comprobar el tiempo que emplea cada una..
>
```

Figura 10. Captura de pantalla la progresión aritmética.

```
> P.geometrica <- function( n=1, a1=1, r=1)
+ {
+ a1 + r^ (n-1)
+ }
> P.geometrica()
[1] 2
> P.geometrica(2,3,2)
[1] 5
> # Otra forma sería
> P. geometrica2<- function(n=1, a1 = 1, r=1)
Error: unexpected symbol in "P. geometrica2"
> P.geometrica2<- function(n=1, a1 = 1, r=1)
+ {
+ if (n>1)
+ {
+ P.geometrica(n-1,a1,r)+r
+ }
+ else
+ {
+ a1
+ }
+ }
> P.geometrica2(2,3,2)
[1] 6
```

Figura 11. Captura de pantalla la progresión geométrica.

```
> # Otra forma sería
> P. geometrica2<- function(n=1, a1 = 1, r=1)
Error: unexpected symbol in "P. geometrica2"
> P.geometrica2<- function(n=1, a1 = 1, r=1)
+ {
+ if (n>1)
+ {
+ P.geometrica(n-1,a1,r)+r
+ }
+ else
+ {
+ a1
+ }
+ }
> P.geometrica2(2,3,2)
[1] 6
> # la última sería
> P.geometrica3<- function(n=1, a1 = 1, r=1)
+ {
+ if (n ==1) return (a1)
+ A = 1:n
+ A[1] = a1
+ for ( i in 2:n) A[i] = A[i-1]*r
+ return(A[n])
+ }
> P.geometrica3(2,3,2)
[1] 6
>
```

Figura 12. Captura de pantalla de otra manera de obtener la progresión geométrica.

```
>
> # Par o Impar
> # Se crean vectores vacíos
> n <- 10
> pares <- c()
> impares <- c()
> for ( i in 1:n) {
+ if ( i %% 2 == 0) pares <- c (pares, i) # es par
+ else impares <- c(impares, i)} # es impar
> pares
[1]  2  4  6  8 10
> impares
[1] 1 3 5 7 9
>
```

Figura 13. Captura de pantalla del proceso de selección de números pares e impares del vector c.

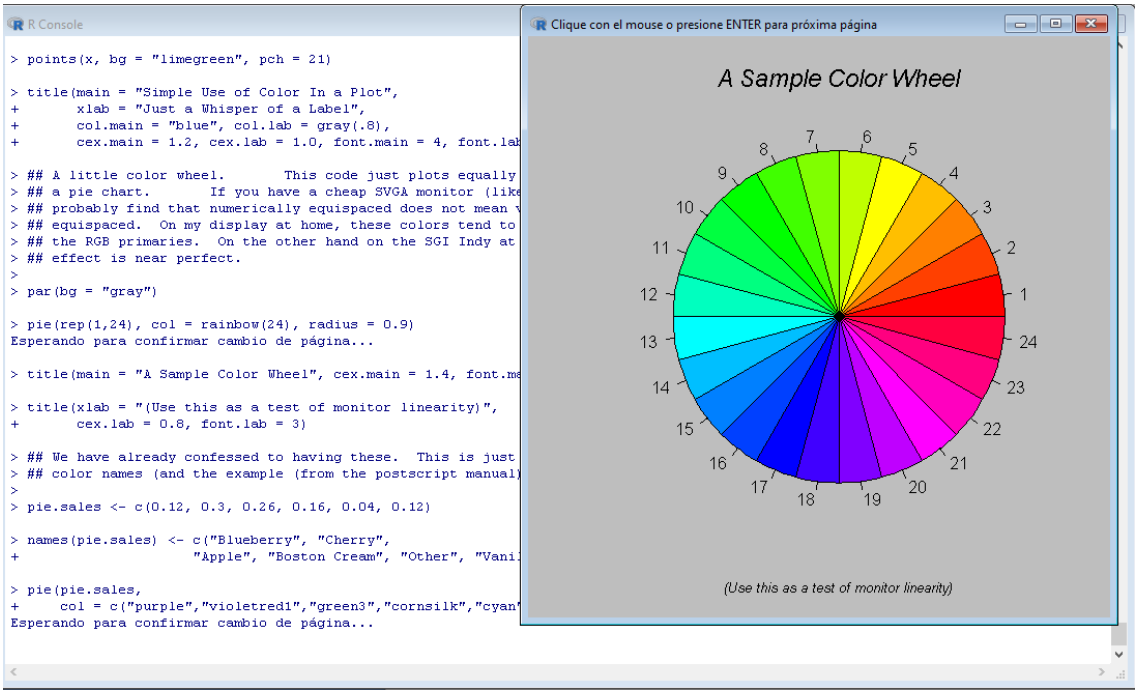


Figura 14. Captura de pantalla del transcurso de la demostración de los gráficos disponibles en R.

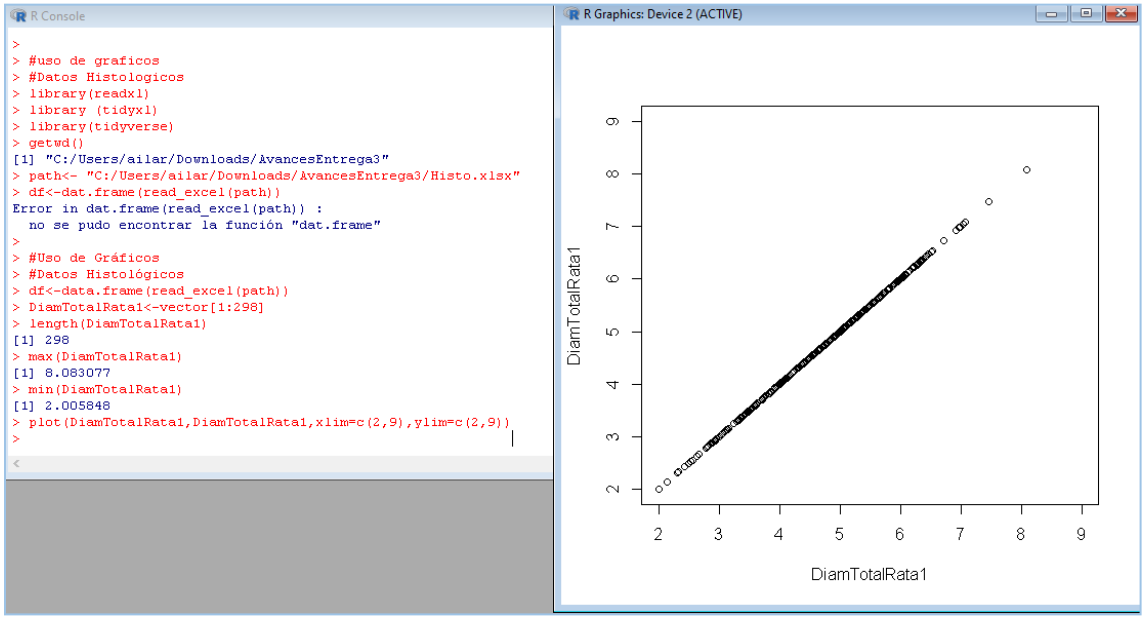


Figura 15. Captura de pantalla del uso de gráfico básico.

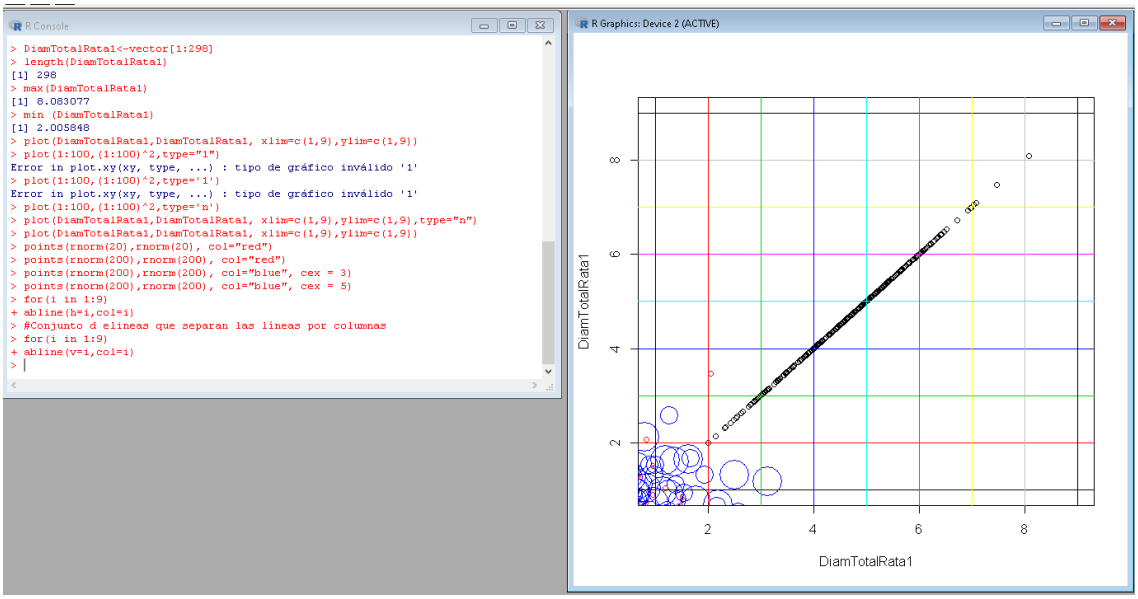


Figura 16. Captura de pantalla del uso de puntos (rojo), puntos más grandes (azul), líneas horizontales y líneas verticales.

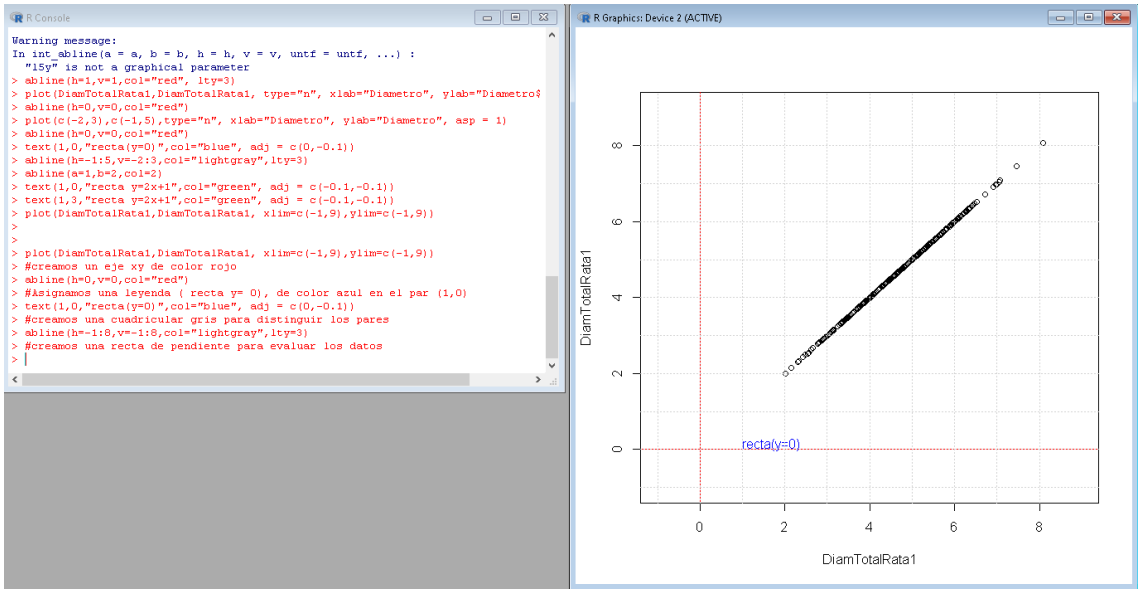


Figura 17. Captura de pantalla del uso de abline() para dibujar una recta en 0.

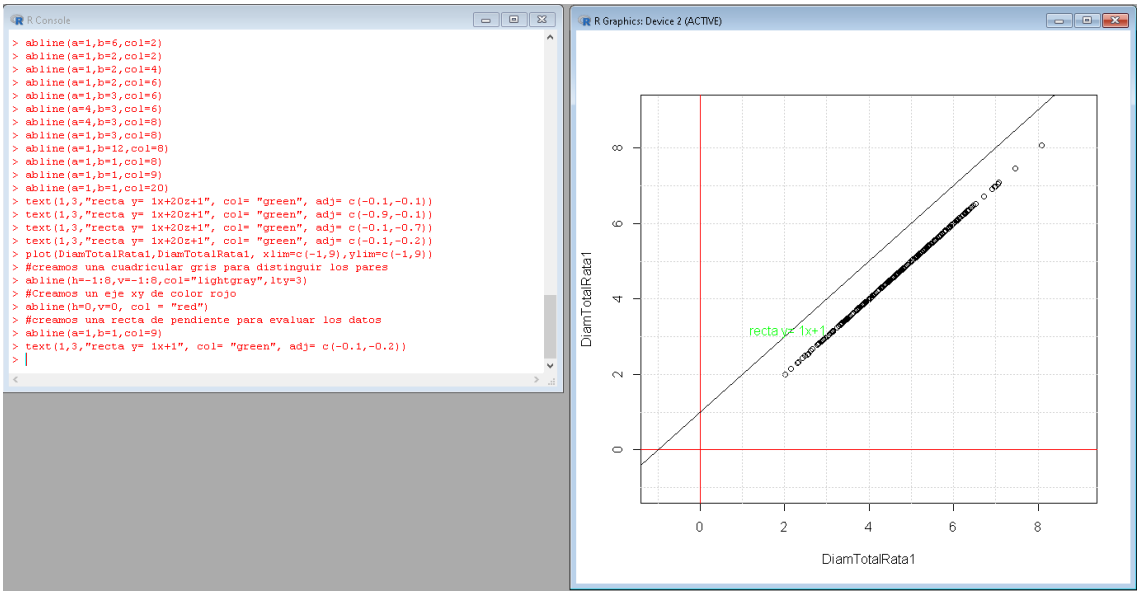


Figura 18. Captura de pantalla del uso de abline() junto con los datos y el texto de la recta(y= 1x + 1) en verde.

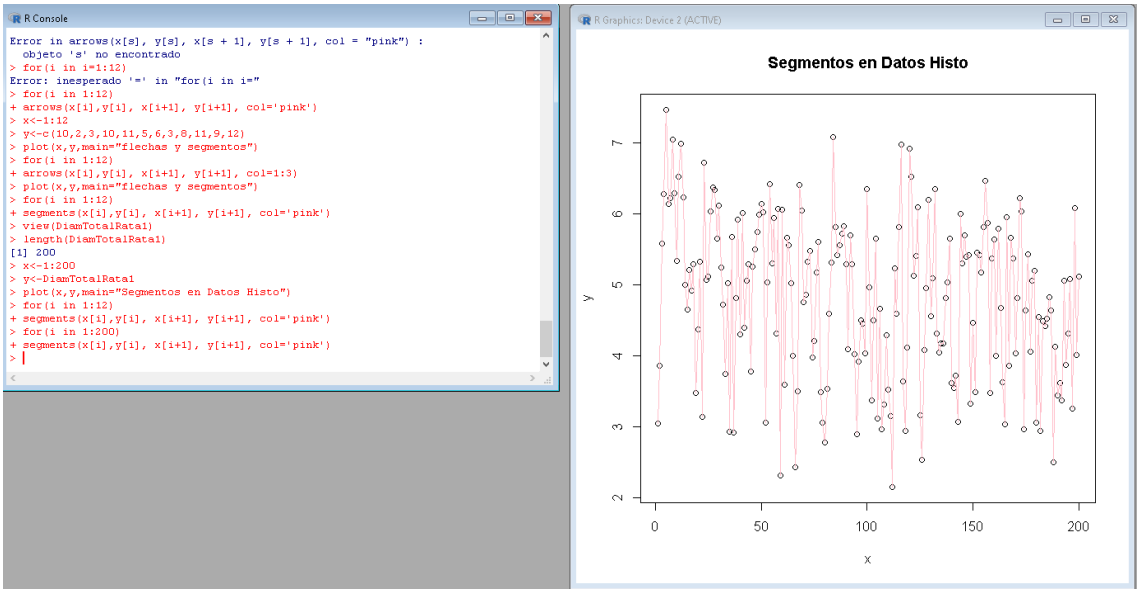


Figura 19. Captura de pantalla del uso de segments() color rosa entre los puntos de los datos.

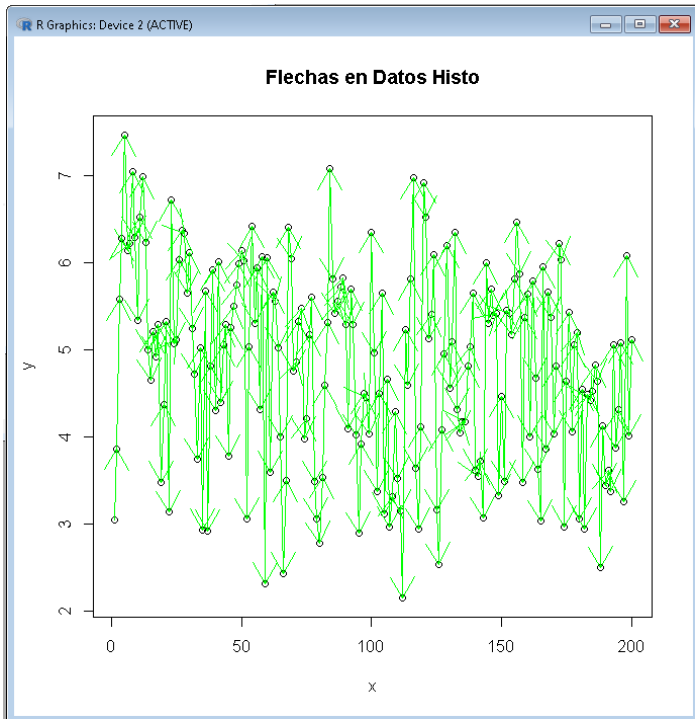


Figura 20. Captura de pantalla del uso de arrows() color verde entre los puntos de los datos.

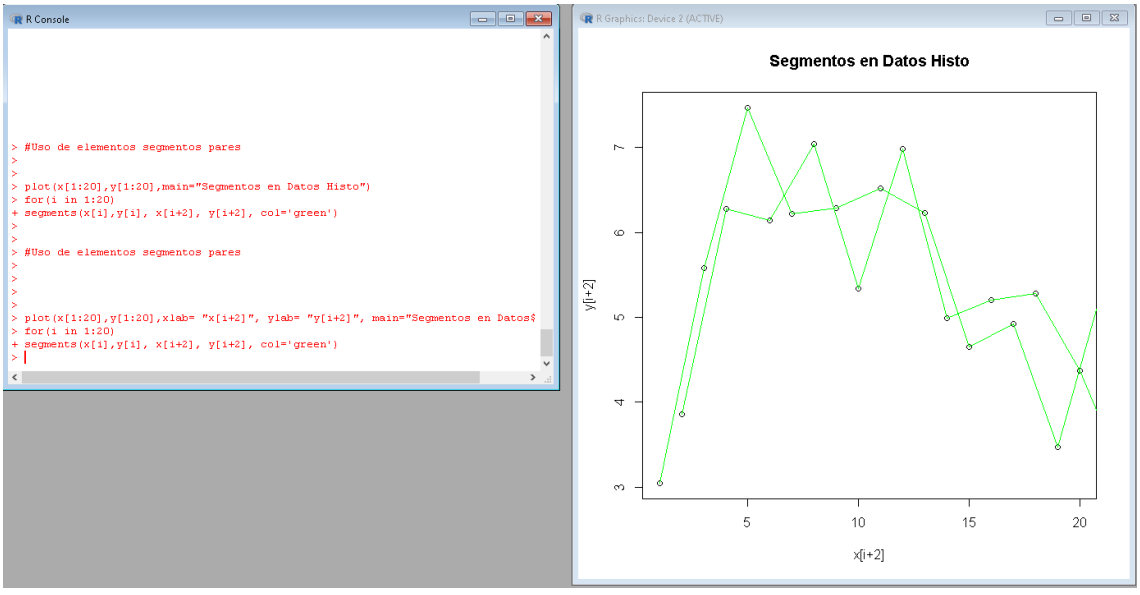


Figura 21. Captura de pantalla del uso de segments() en elementos pares e impares dentro de un conjunto de datos.

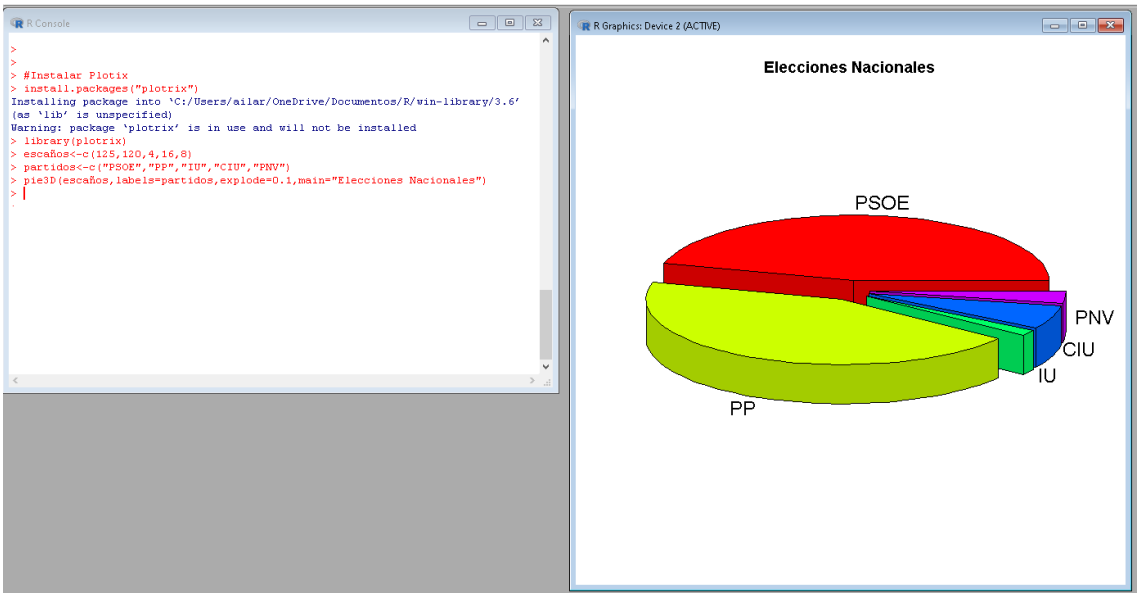


Figura 22. Captura de pantalla del uso de pie3D().

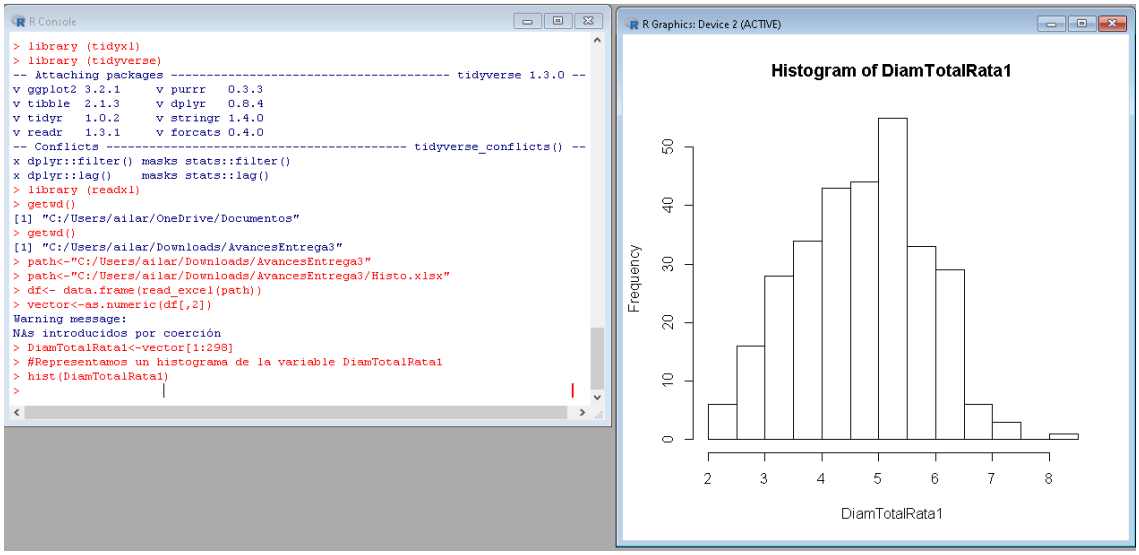


Figura 23. Captura de pantalla del uso de hist() con datos histológicos.

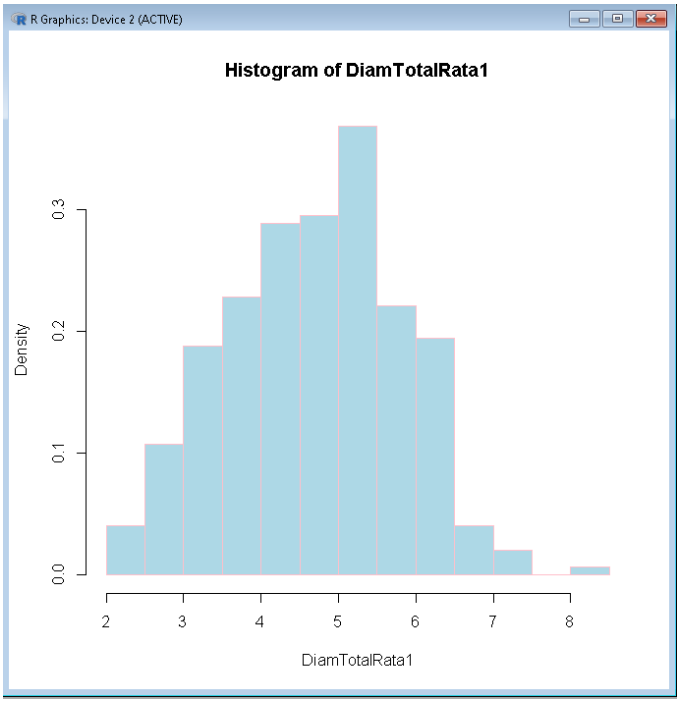
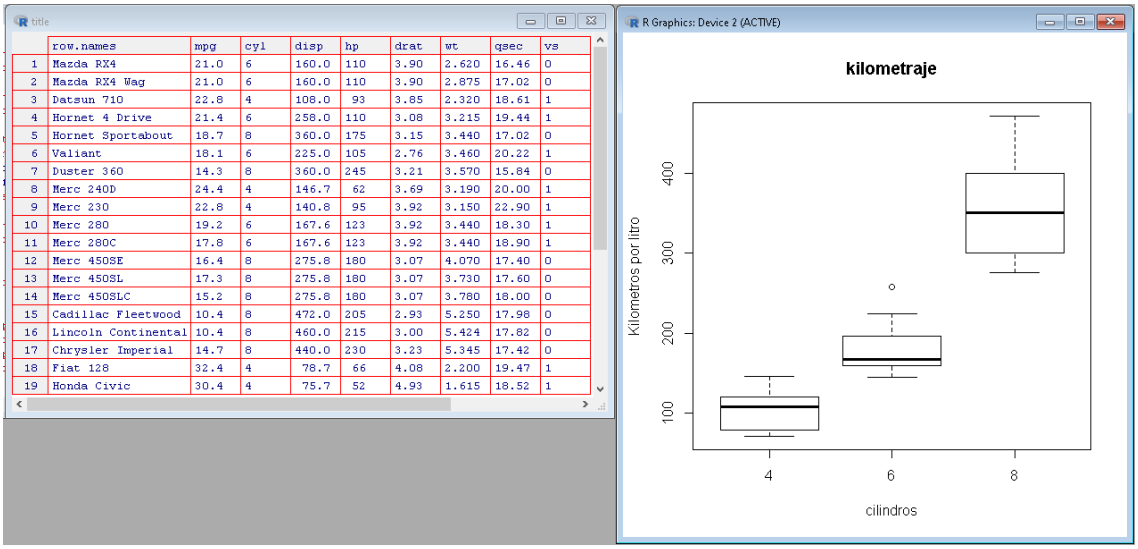


Figura 24. Histograma con 13 intervalos (12 puntos de corte) de datos histológicos, de color azul y líneas rosas.



	row.names	mpg	cyl	disp	hp	drat	wt	qsec	vs
1	Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
2	Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
3	Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
4	Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
5	Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0
6	Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1
7	Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0
8	Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1
9	Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1
10	Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1
11	Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1
12	Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0
13	Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0
14	Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0
15	Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0
16	Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0
17	Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0
18	Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1
19	Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1

Figura 25. Captura de pantalla de boxplot() de los datos mtcars.

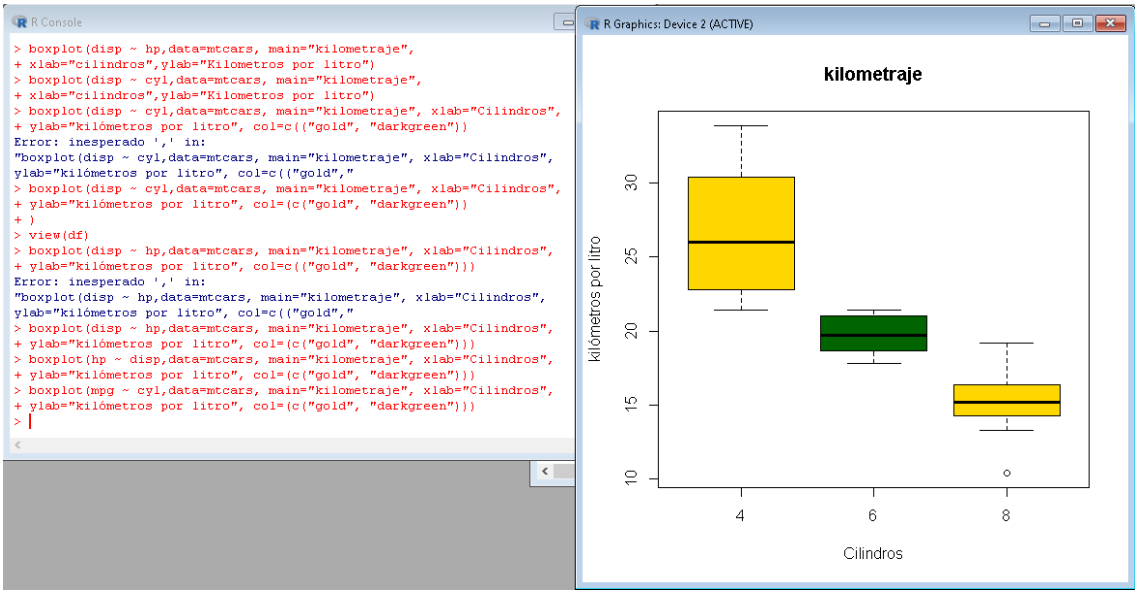


Figura 26. Captura de pantalla de boxplot() modificando las cajas por individual.

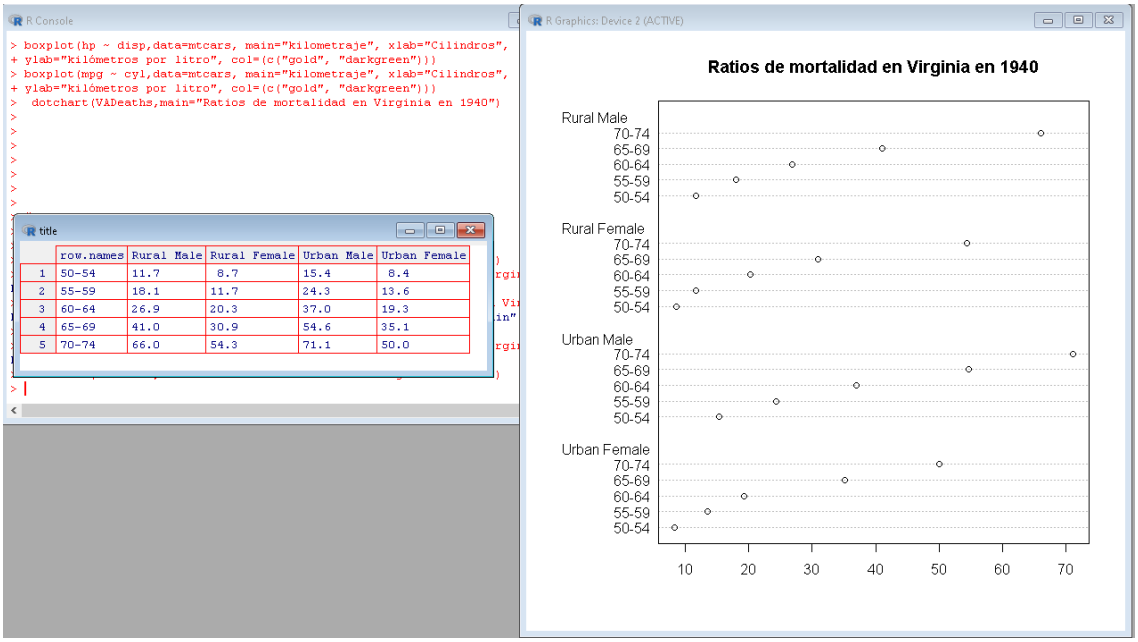


Figura 27. Captura de pantalla de dotchart() para la base de datos VADeaths de R.

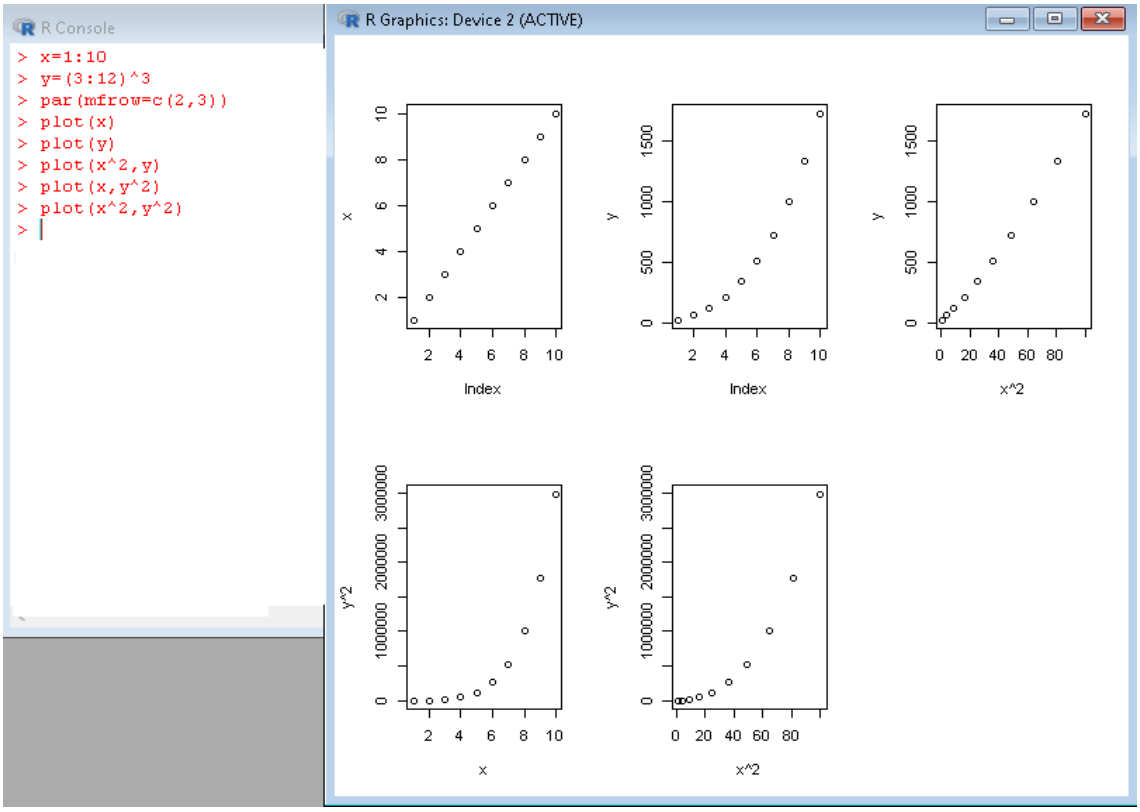


Figura 28. Captura de pantalla de par() y ubicación de los plot() con argumento mfrow.

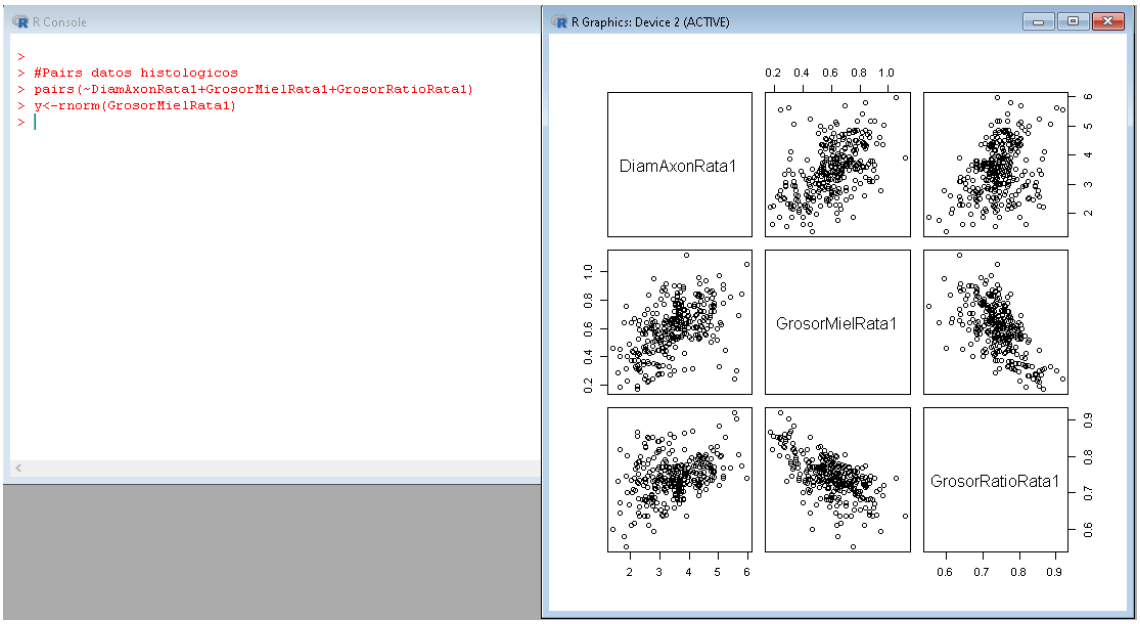


Figura 29. Captura de pantalla de la función pairs() con datos histológicos.

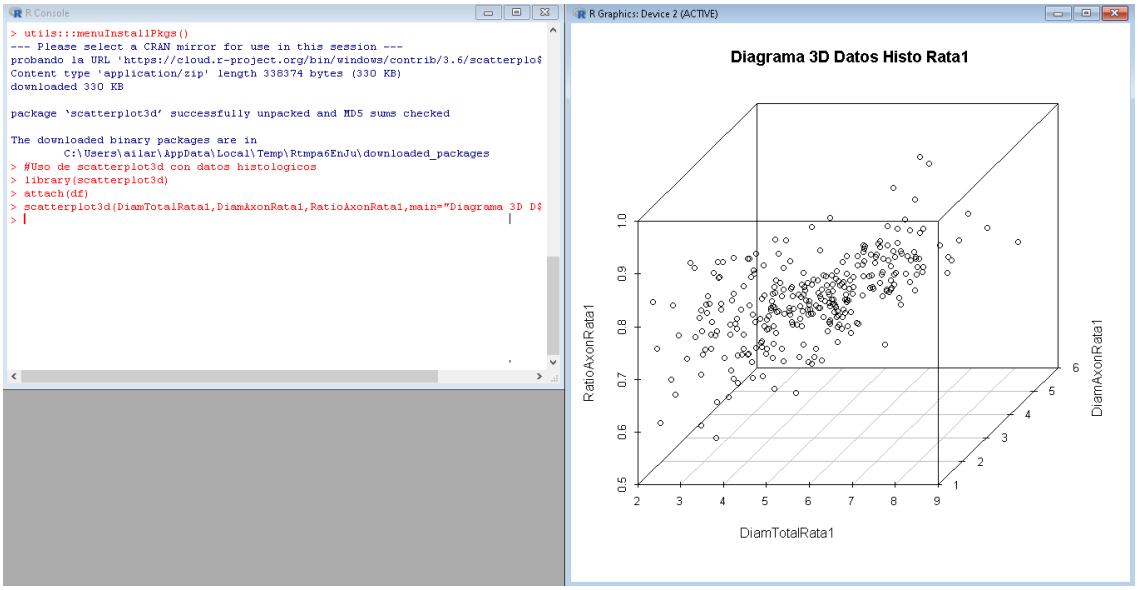


Figura 30. Captura de pantalla de la función attach() y scatterplot3d() con datos histológicos.

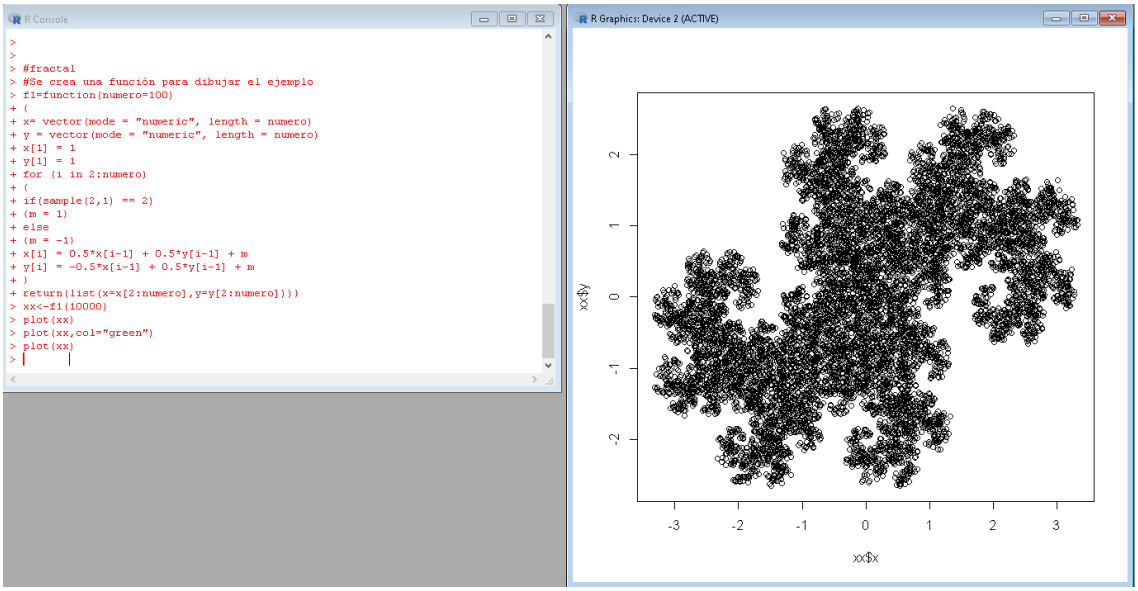


Figura 31. Captura de pantalla de la función fractal f1(puntos) que se crea con los elementos que tiene R, se pueden corroborar en la Tabla I.

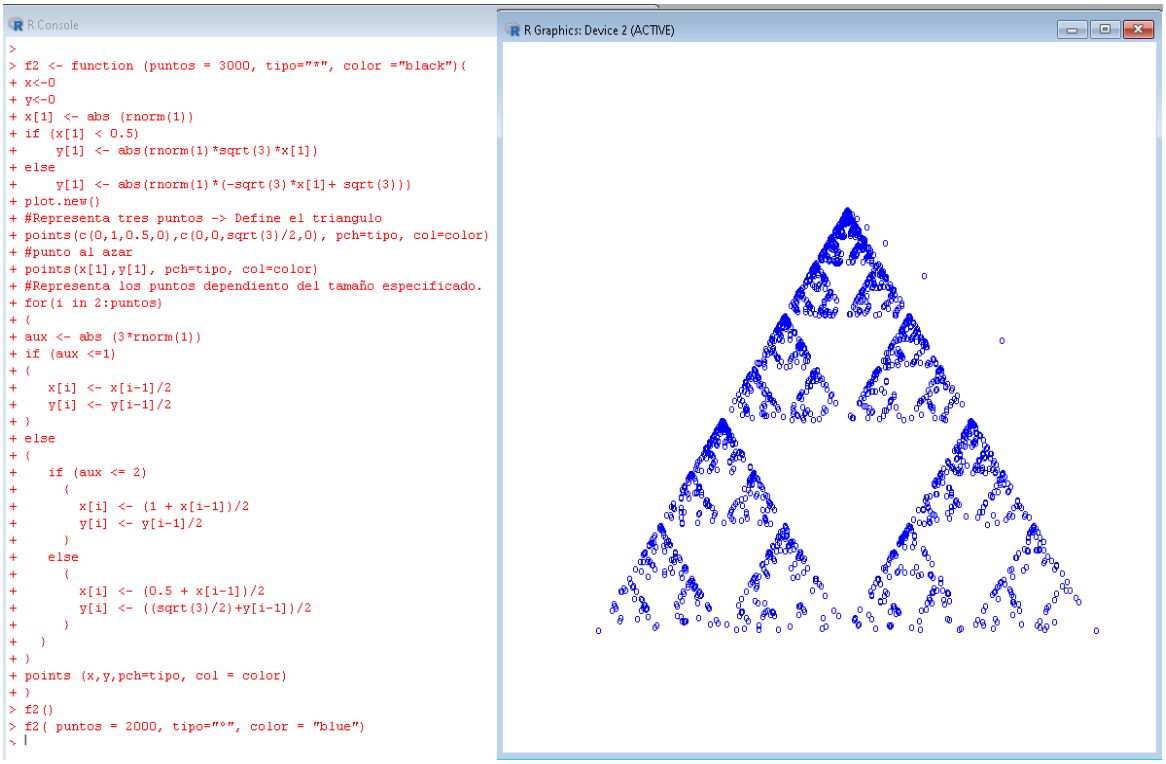


Figura 32. Captura de pantalla de la función fractal f2(puntos, tipo, color) que se crea con los elementos que tiene R, se pueden corroborar en la Tabla I.

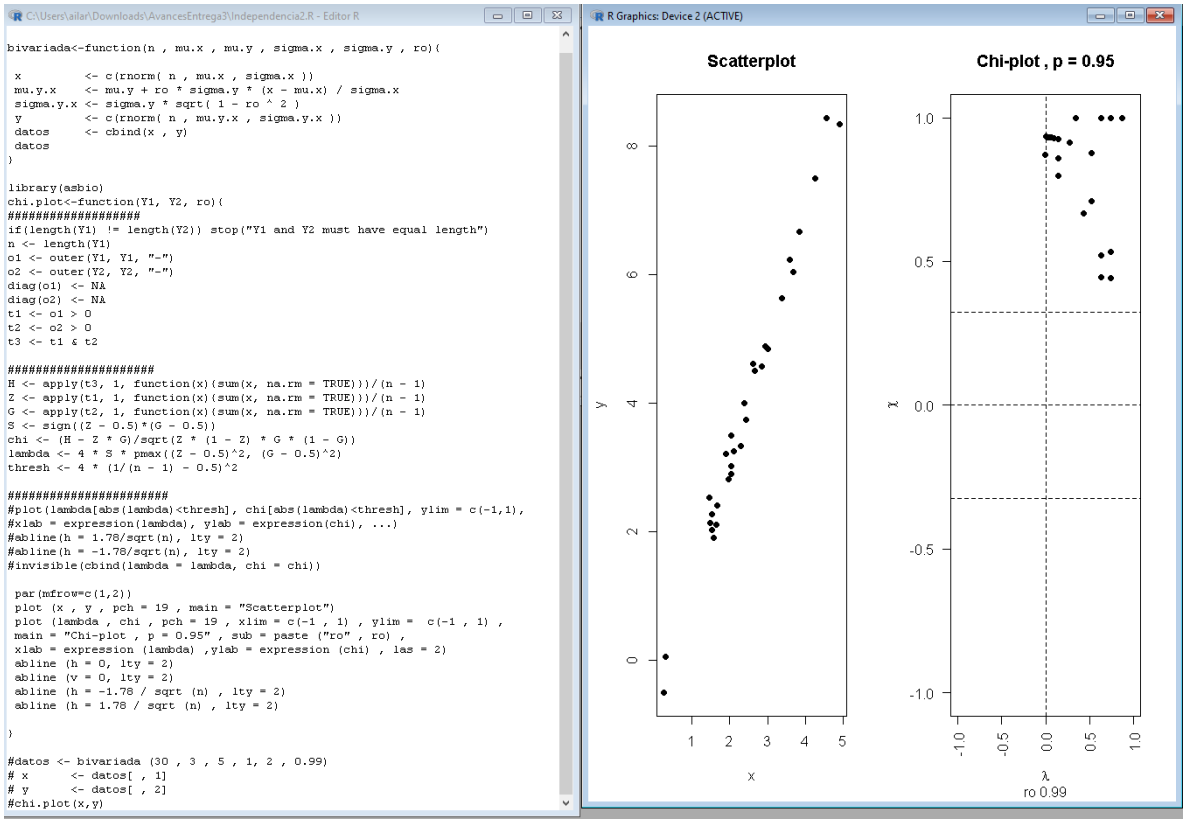


Figura 33.Captura de pantalla de la función de chi.plot y la obtención de parámetros involucrados en la Ecuación 1–Ecuación 4.

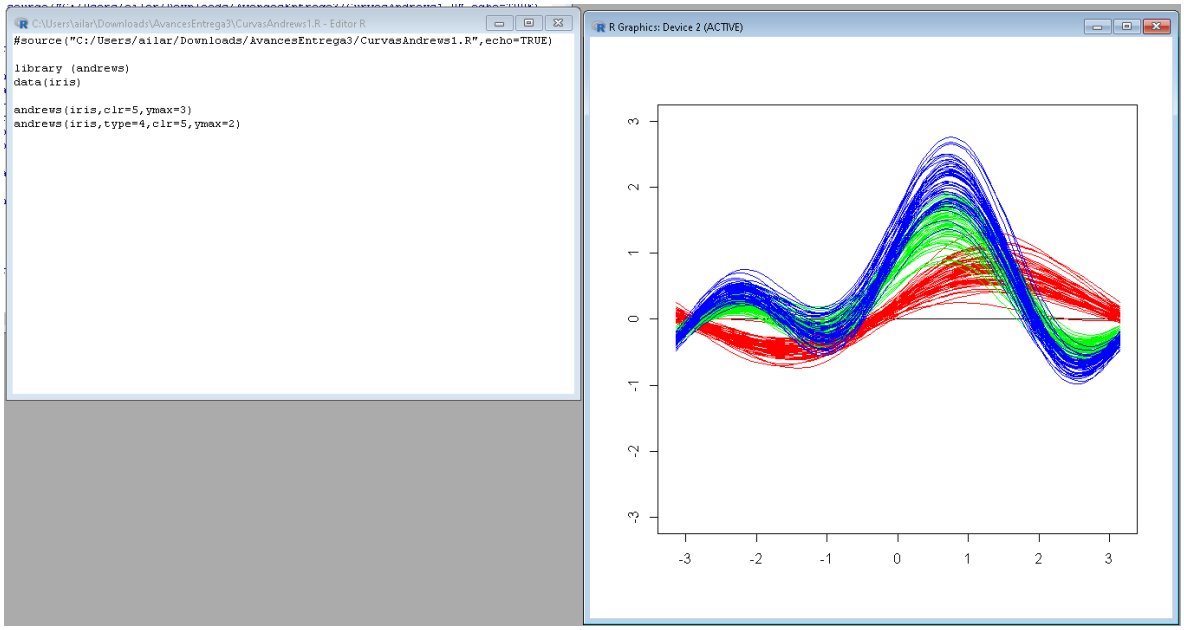


Figura 34. Captura de pantalla de la función andrews() y la respuesta de salida.

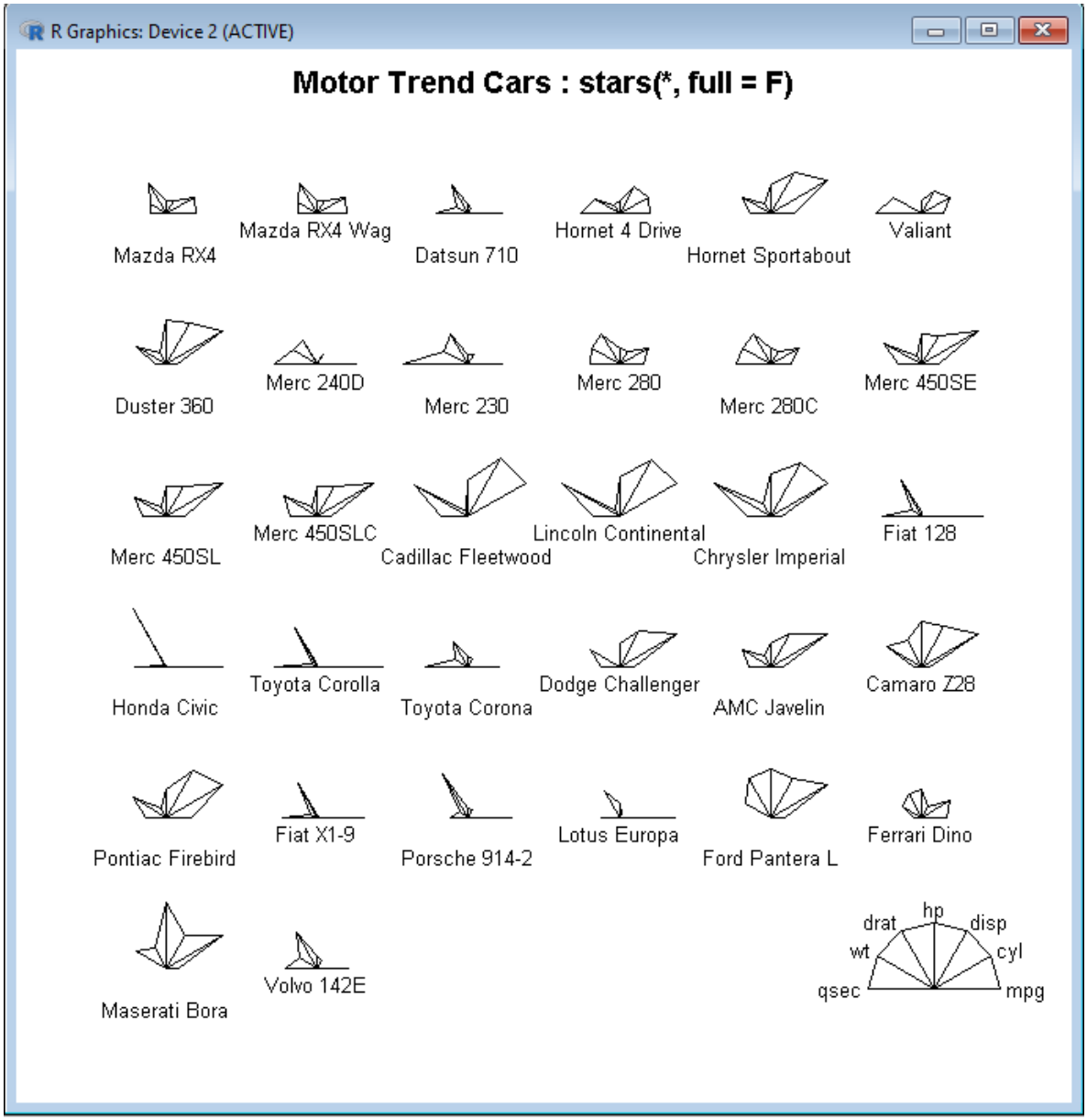


Figura 35. Captura de pantalla de la función stars() con la base de datos mtcars.

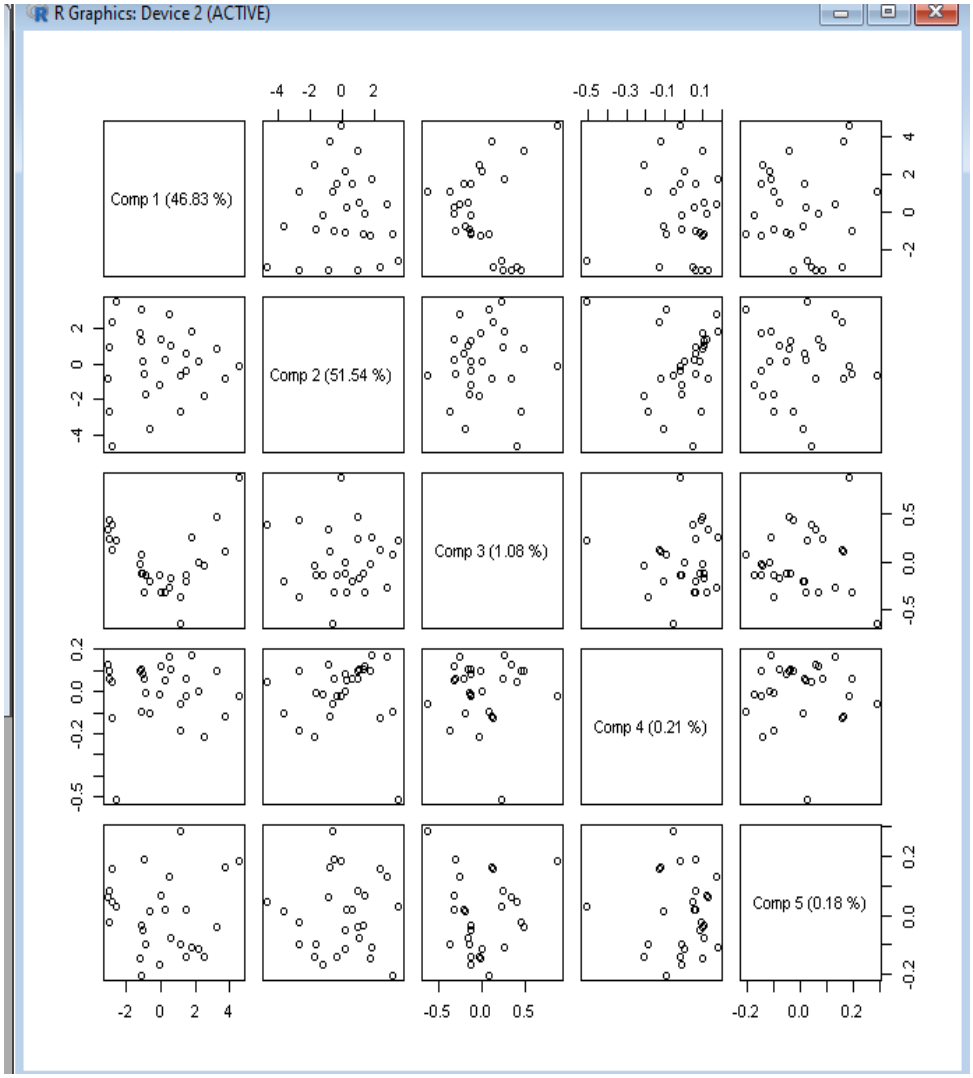


Figura 36. Captura de pantalla del score plot de la base de datos yarn que corresponde a 268 longitudes de onda de un espectro.

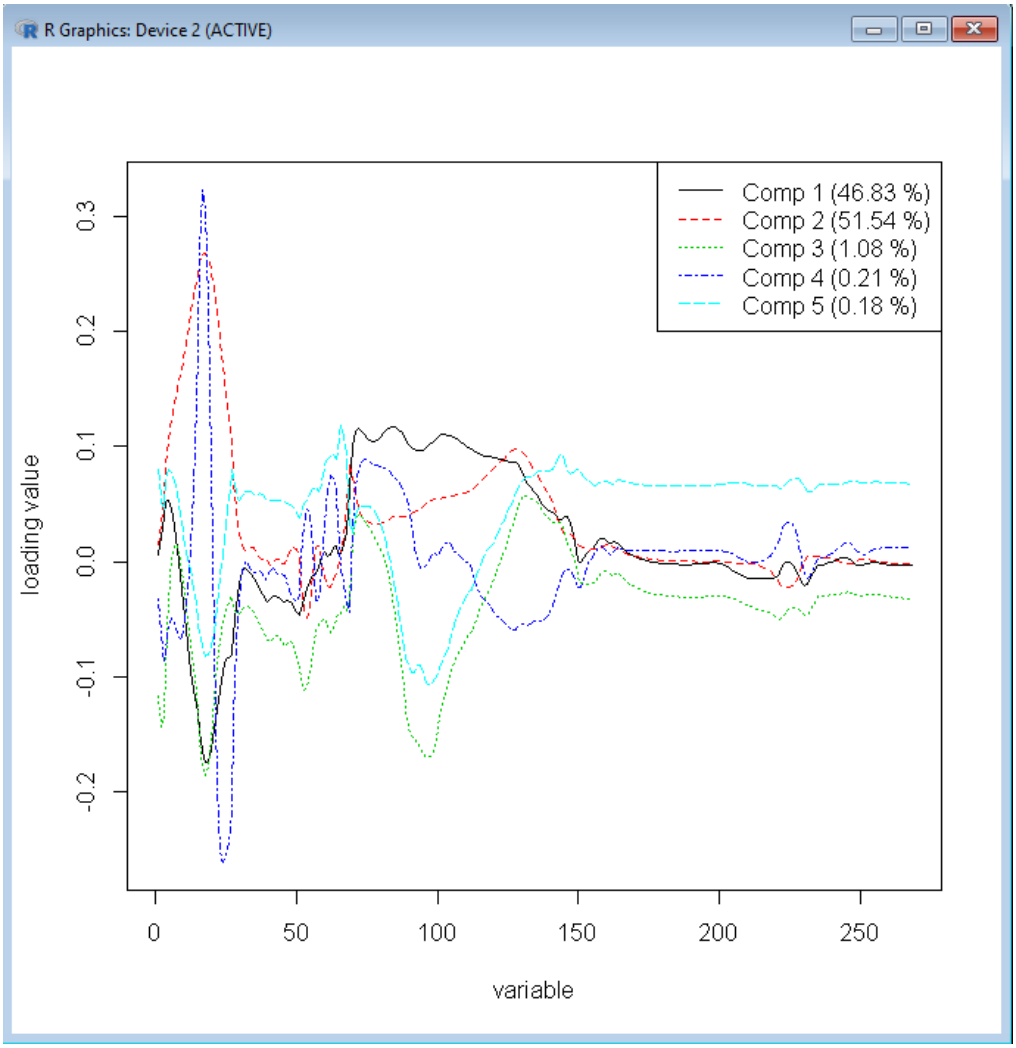


Figura 37. Captura de pantalla de la lectura de las componentes principales del espectro de la base de yarn y sus densidades.

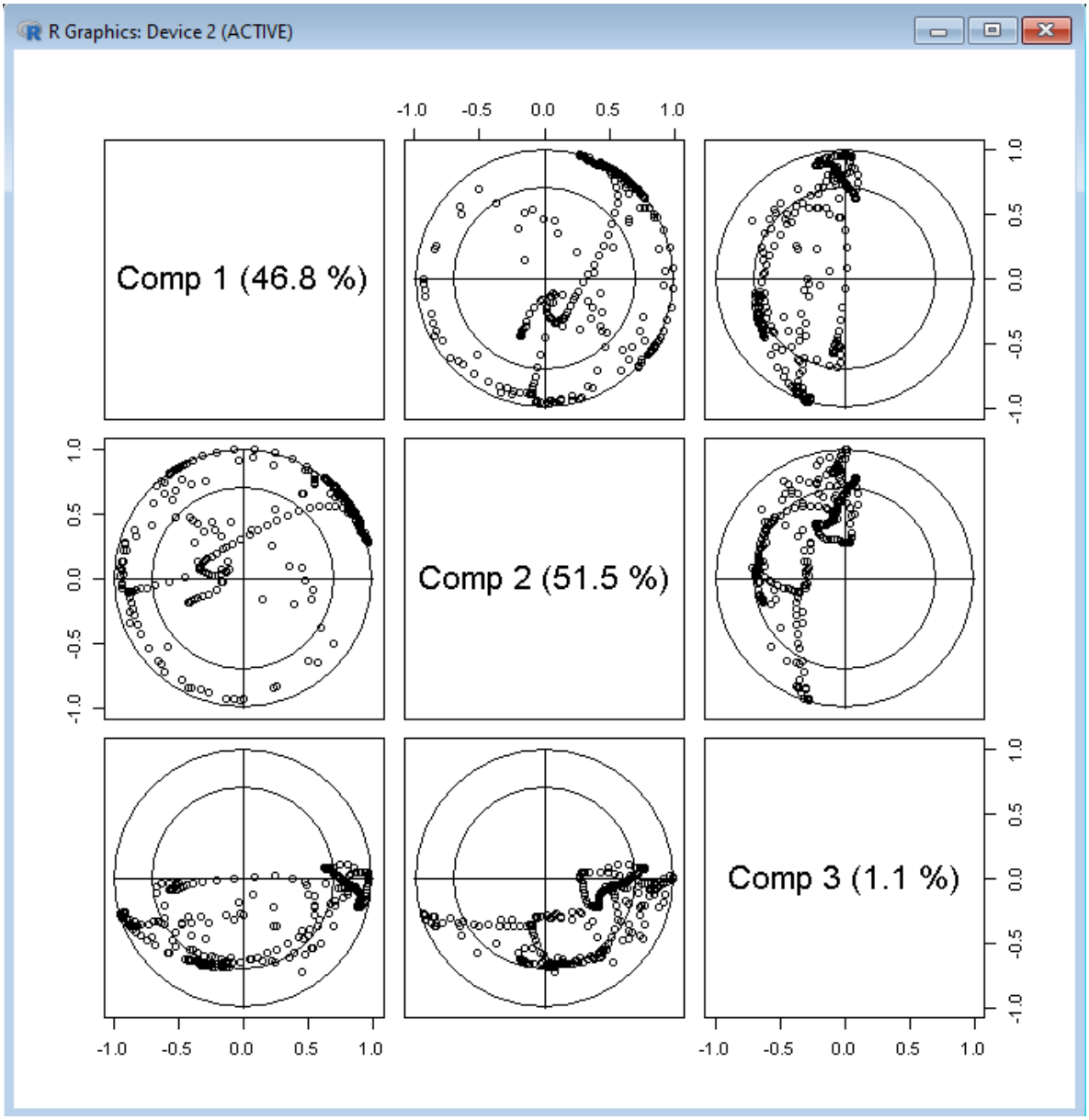


Figura 38. Captura de pantalla de las correlaciones entre las diferentes componentes principales de la base de datos yarn.

ECUACIONES

$$H_i = \sum_{j \neq i} I(x_j \leq x_i, y \leq y_i)/(n-1)$$ Ecuación 1

$$F_i = \sum_{j \neq i} I(x_j \leq x_i)/(n-1)$$ Ecuación 2

$$G_i = \sum_{j \neq i} I(y \leq y_i)/(n-1)$$ Ecuación 3

$$S_i = sign\left\{\left(F_i - \frac{1}{2}\right)\left(G_i - \frac{1}{2}\right)\right\}$$ Ecuación 4

$$\chi_i = (H_i - F_i G_i)/\{F_i(1 - F_i)G_i(1 - G_i)\}^{1/2}$$ Ecuación 5

$$\Lambda_i = 4S_i max\left\{(F_i - \frac{1}{2})^2, (G_i - \frac{1}{2})^2\right\}$$ Ecuación 6