Gráficos Estadísticos con R

Juan Carlos Correa y Nelfi González
Posgrado en Estadística
Universidad Nacional-Sede Medellín

e-mail: jccorrea@perseus.unalmed.edu.co

2002

Índice General

Prefacio

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 muy 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.

Estas notas presentan gran variedad de gráficos de uso corriente en la disciplina estadística y que han probado ser efectivos en la representación de datos. Muchos ejemplos aquí presentados están basados en los aportes de diferentes usuarios que han publicado sus rutinas en la lista r-help del CRAN, a los cuales queremos hacer un merecido reconocimiento.
Esperamos que el lector aprecie el poder de una buena gráfica, tanto como los que trabajamos en el área estadística y que la lectura de este documento le ayude a utilizar R en su elaboración.

Capítulo 1

Generalidades

"Un gráfico puede valer más que mil palabras,
pero puede tomar muchas palabras para hacerlo"
John Tukey

1.1 Introducción

La presentación de datos mediante gráficos es algo que se realiza a diario y en forma casi natural por personas de las más diferentes profesiones. La revista americana *LIFE* tenía como consigna "Una foto vale más que mil palabras". La capacidad de visualización del hombre hace que esto sea casi cierto. En comparación con otras formas de presentación de los datos, los gráficos nos permiten, de una mirada, comprender el comportamiento de los datos, aún de datos muy complejos, por lo tanto ahorran tiempo al analista de información. Los gráficos estadísticos nos permiten usar nuestra habilidad para visualmente procesar información de un gráfico. Esto nos permite hacer juicios respecto a la variabilidad, escala, patrones y tendencias de los datos.

Wainer (1990) dice:

> "En más de 200 años desde el desarrollo inicial de las técnicas gráficas, hemos adquirido algún conocimiento en sus usos, nacida de la experiencia."

Es desafortunado el poco énfasis que la mayoría de textos en estadística ponen en la parte gráfica. Unos pocos (Moore, 1979; Campbell, 1990) hacen énfasis en los errores de interpretación en la presentación de gráficos en los

medios de comunicación, pero la gran mayoría, en especial los que se utilizan como texto de clase, sólo presentan algunos gráficos más como un material extra que como una herramienta fundamental en el trabajo aplicado.

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

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 (1990) 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" los divide en dos categorías:

1. Un *gráfico fuertemente bueno* muestra todo lo que queremos conocer sólo con mirarlo.

2. Un *gráfico débilmente bueno* nos muestra lo que necesitamos conocer observándolo, una vez sepamos como mirarlo.

Una buena descripción puede transformar un gráfico débilmente bueno en uno fuertemente bueno. Debemos siempre buscar esta transformación cuando sea posible. Una buena descripción informa al lector y obliga al que produce el gráfico a pensar porqué y cómo está presentando el gráfico.

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.

1.1.1 Cuándo presentar los datos ?

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 torta (pie). Si el conjunto de datos es menor de 20, estos deben presentarse en una tabla, ya que usualmente la tabla supera al gráfica.

PRINCIPIO: Si usted debe aclarar un gráfico con los números, utilice una tabla.

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

Tufte presenta un índice para medir la cantidad de información irrelevante en un gráfico:

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

Este índice también se puede interpretar como el porcentaje de tinta del gráfico que no puede ser borrado sin afectar los datos. Wainer (1990), en una crítica al índice de Tufte, propugna por no sólo la eficiencia en la transmisión de los datos, sino también por la elegancia.

1.1.2 Equipo y software para gráficos

Un gráfico estadístico es una representación visual de datos estadísticos. Los datos estadísticos son observaciones o funciones de una o más variables. La elaboración de un buen gráfico exige la utilización de software y equipo de alta calidad. A nivel de equipos necesitamos un computador con un buen monitor y una impresora excelente. Nadie puede pretender obtener buenos gráficos con una impresora de matriz de puntos, cuando para ello se requiere de un plotter o de una impresora láser. Un buen software para gráficos aprovecha al máximo el equipo disponible, pero no puede hacer más de lo que el equipo permita. Existen muchos programas con poderosos módulos gráficos incorporados.

Lamentablemente la mayoría de los usuarios se limitan a elaborar gráficos de paquete, esto es, gráficos cuya presentación estética es agradable y pasa a ser la parte más importante del mismo. Tal vez esto se debe al impacto tan profundo que han tenido los medios de comunicación, en especial la de ciertos

formatos para llamar la atención del desprevenido parroquiano. Los gráficos de paquete aparecen corrientemente en hojas electrónicas de cálculo y algunos paquetes estadísticos. La característica de ellos es la poca flexibilidad y la poca libertad que dan al usuario para adaptarlos a sus necesidades. Ningún programa reemplaza al analista en el diseño de sus gráficos, ni puede adivinar que pretende esta persona mostrar.

1.2 Percepción gráfica

Cleveland and McGill (1984, 1985, 1987) 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:

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. Area.
6. Volumen, densidad y saturación de color.
7. Escala de color.

Algunos de ellos pueden parecer obvios, pero otros no.

1.3 Propósito de un gráfico

Para qué un gráfico? Existen razones que el analista debe esgrimir para justificar la realización de un gráfico. Fienberg (1979) refiere a una clasificación de los motivos para presentar un gráfico:

- **Gráficos de Propaganda**: Ellos intentan mostrar lo que ya se ha aprendido por otras técnicas.

- **Gráficos Analíticos**: Estos gráficos nos permiten ver lo que puede estar ocurriendo.
- **Gráficos Sustitutos de Tablas**: En estos gráficos hay que leer los números que contienen.
- **Gráficos para Decoración**: Estos gráficos se presentan porque son bonitos.

Como en toda actividad estadística la presentación de un gráfico debe tener consideraciones de carácter ético, Burns (1993) enuncia estos principios:

- Entendibilidad
    1. Nos permite el gráfico las relaciones entre las variables?
    2. Interactúan los elementos en el gráfico para maximizar nuestra percepción de las relaciones entre las variables?
- Claridad
    1. Son los elementos del gráfico claramente distinguibles?
    2. Son los elementos más importantes del gráfico visualmente prominentes?
- Consistencia
    1. Son los elementos de los gráficos consistentes con su uso en gráficos anteriores?
    2. Existen nuevos elementos del gráfico que requieren una descripción adicional?
- Eficiencia
    1. Están los elementos del gráfico eficientemente representando los datos?
    2. Hay elementos en el gráfico que sirven a más de un propósito?
- Necesidad
    1. Es el gráfico una forma útil de representar estos datos?

2. Es cada elemento en el gráfico necesario?

- Confiabilidad

1. Están los datos adecuadamente colocados en la región de datos?
2. Están los datos representados adecuadamente por la escala?

1.4 Elementos de un Gráfico

- Título Principal
- Título Secundario o Subtítulo
- Descripción del Gráfico
- Región de Datos y Símbolos
- Eje Horizontal y Escala
- Eje Vertical y Escala
- Apuntadores
- Descriptores de Señales y marcas

1.5 Recomendaciones Generales para Elaborar un Gráfico

- Haga que sus datos sobresalgan. Evite lo superfluo.
- Utilice elementos prominentes para mostrar sus datos.
- Utilice un par de líneas por cada variable. Haga que el interior del rectángulo formado por las líneas de escala sean la región de sus datos. Coloque marcas afuera de la región de los datos.
- No apeñuzque la región de datos.
- No exagere el número de marcas.

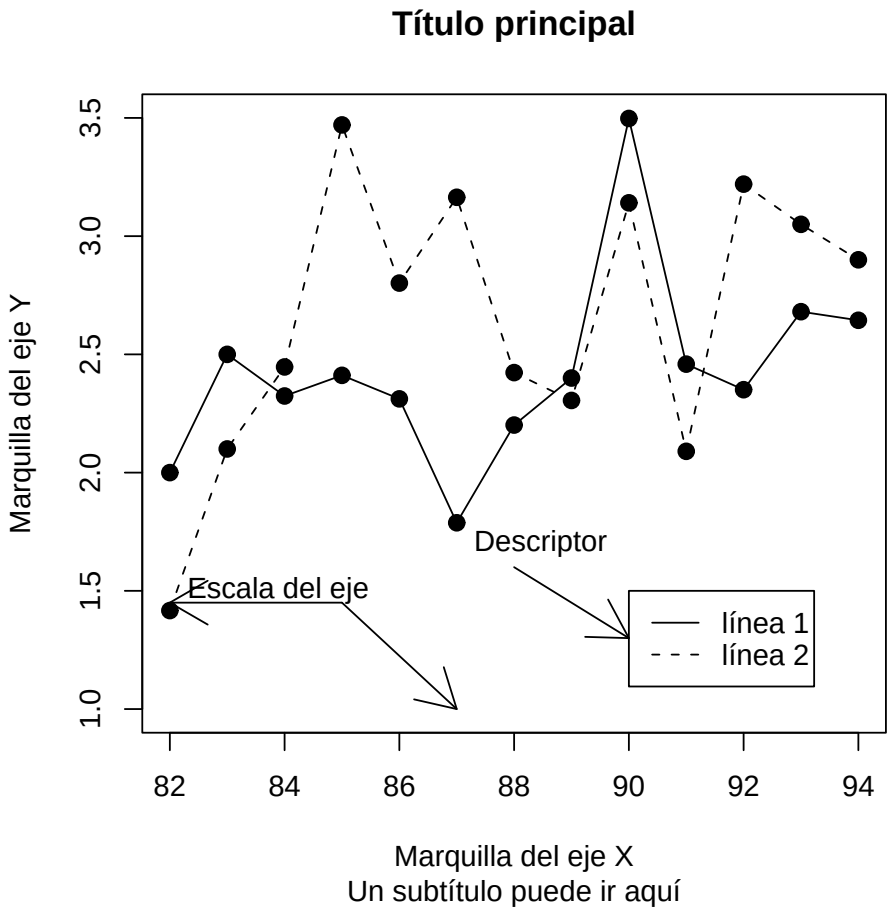

Figura 1.1: *En esta parte va una descripción del gráfico. Esta explicación no debe ser una repetición de la información mostrada en el gráfico.*

- Utilice una línea de referencia cuando haya un valor importante que deba verse a través de todo el gráfico, pero no deje que interfiera con sus datos.
- No permita que las marquillas o referencias en la región de datos interfieran con los datos cuantitativos o que se amontonen.
- Evite colocar notas, marcas o señales a un lado de la región de datos. Coloque notas en el texto o la explicación.
- Gráficos sobrepuestos deben ser visualmente distinguibles.
- Datos sobrepuestos deben ser visualmente discernibles.
- La claridad visual debe conservarse bajo reducción y reproducción del gráfico.

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

- mostrar los datos
- inducir al observador a pensar acerca de la sustancia en lugar de la metodología, el diseño gráfico, la tecnología que produjo el gráfico, o en algo más
- evitar la distorción en el mensaje de los datos
- presentar muchos números en un pequeño espacio
- hacer que grandes conjuntos de datos tengan coherencia
- inducir a los ojos a comparar diferentes partes de los datos
- revelar diferentes detalles de los datos, desde la perspectiva global a los detalles particulares
- tener un propósito razonablemente claro: la descripción, la exploración, la tabulación, o la decoración
- estar muy integrado a las descripciones estadísticas y verbales del conjunto de datos.

1.6 Gráficos distorsionados

Tufte dice "un gráfico no distorsiona si la representación visual de los datos es consistente con la representación numérica".

Experimentos sobre la percepción de áreas de círculos sugiere que en el común de las personas es menor al área real siguiendo la fórmula

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

donde AP es el Area Percibida, AR es el Area Real y α es 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 divertir al lector y no un medio de comunicación efectivo de los datos. Esas persona 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.

1.7 Pecados comunes en gráficos técnicos

Los gráficos técnicos usualmente no presentan problemas de exceso de decoración, pero 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 2

Gráficos Estadísticos Univariables en R

2.1 Elementos graficadores en R

2.1.1 Funciones para Especificar Devices

- postscript(archivo, comando, horizontal=F, width, height, rasters, pointsize=14, font=1, preamble=ps.preamble, fonts=ps.fonts)

- printer(width=80, height=64, file=" ", command=" ") show()

2.1.2 Algunos Parámetros para Graficar en R

log='<x|y|xy>'	Ejes Logarítmicos
main='título'	
new=<logical>	Adiciona sobre el gráfico actual
sub='título de abajo'	
type='<l|p|b|n>'	Línea, puntos, ambos, ninguno
lty=n	Tipo de Línea
pch='.'	Caracter de dibujo
xlab='Nombre del eje x'	
ylab='Nombre del eje y'	
xlim=c(x_{minimo}, x_{maximo})	
ylim=c(y_{minimo}, y_{maximo})	

2.1.3 Gráficos Unidimensionales

- barplot(Estaturas)
- barplot(Estaturas, amplitud, nombres, space=.2, inside=TRUE, beside=FALSE, horiz=FALSE, legend, angle, density, col, blocks=TRUE)
- boxplot(..., rango, amplitud, varwidth=FALSE, notch=FALSE, names, plot=TRUE)
- hist(x, nclass, breaks, plot=TRUE, angle, density, col, inside)

2.1.4 Gráficos Bidimensionales

- lines(x, y, type="l")
- points(x, y, type="p"))
- plot(x, y, type="p", lty=1:5, pch=, col=1:4)
- points(x, y, type="p", lty=1:5, pch=, col=1:4)
- lines(x, y, type="l", lty=1:5, pch=, col=1:4)
- plot(x, y, type="p", log=)
- abline(coef)
- abline(a, b)
- abline(reg)
- abline(h=)
- abline(v=)
- qqplot(x, y, plot=TRUE)
- qqnorm(x, datax=FALSE, plot=TRUE)

2.1.5 Gráficos Tridimensionales

- contour(x, y, z, v, nint=5, add=FALSE, labels)
- interp(x, y, z, xo, yo, ncp=0, extrap=FALSE)
- persp(z, eye=c(-6,-8,5), ar=1)

2.1.6 Múltiple Gráficos por Página (Ejemplo)

```
par(mfrow=(nrow, ncol), oma=c(0, 0, 4, 0))
mtext(side=3, line=0, cex=2, outer=T,"Este es
un Titulo para Toda la Página")
```

2.2 Gráficos Univariables

2.2.1 Árboles de Tallo y Hoja

Este gráfico fue propuesto por Tukey (1977) y a pesar de no ser un gráfico para presentación definitiva se utiliza a la vez que el analista recoge la información ve la distribución de los mismos. Estos gráficos son fáciles de realizar a mano y se usan como una forma rápida y no pulida de mirar los datos.

Qué nos muestra?

1. El centro de la distribución.
2. La forma general de la distribución

   Simétrica Si las porciones a cada lado del centro son imágenes espejos de las otras.

   Sesgada a la izquierda Si la cola izquierda (los valores menores) es mucho más larga que los de la derecha (los valores mayores)

   Sesgada a la derecha Opuesto a la sesgada a la izquierda.
3. Desviaciones marcadas de la forma global de la distribución.

   Outliers Observaciones individuales que caen muy por fuera del patrón general de los datos.

   gaps Huecos en la distribución

Ventajas de gráfico:

1. Muy fácil de realizar y puede hacerse a manos.

Las desventajas son:

1. El gráfico es tosco y no sirve para presentaciones definitivas.
2. Funciona cuando el número de observaciones no es muy grande.
3. No permite comparar claramente diferentes poblaciones

```
7 67
7 11
6 8
6 00
5 5666677789999
5 000000011233344444
4 5555555666777788889999
4 00334
3 88
  ----+----+----+----+--

 Datos de velocidades registradas con radar
    Fecha de observacion: Sept. 10, 1994
```

2.2.2 Boxplot o Caja de Tukey

Este ha sido un aporte fundamental realizado por Tukey (1977). Es un gráfico simple, ya que se realiza básicamente con cinco números, pero poderoso. Se observa de una forma clara la distribución de los datos y sus principales características. Permite compara diversos conjuntos de datos simultáneamente.

Como herramienta visual se puede utilizar para ilustrar los datos, para estudiar simetría, para estudiar las colas, y supuestos sobre la distribución, también se puede usar para comparar diferentes poblaciones.

Este gráfico contiene un rectángulo, usualmente orientado con el sistema de coordenadas tal que el eje vertical tiene la misma escala del conjunto de datos. La parte superior y la inferior del rectángulo coinciden con el tercer

cuartil y el primer cuartil de los datos. Esta caja se divide con una línea horizontal a nivel de la mediana. Se define un "paso" como 1.5 veces el rango intercuartil, y una línea vertical (un bigote) se extiende desde la mitad de la parte superior de la caja hasta la mayor observación de los datos si se encuentran dentro de un paso. Igual se hace en la parte inferior de la caja Las observaciones que caigan más allá de estas líneas son dibujadas individualmente. La definición de los cuartiles puede variar y otras definiciones de el paso son planteadas por otros autores (Frigge et al., 1989).

Propiedades del gráfico de caja

1. Cinco números de resumen de los datos son representados gráficamente de tal forma que proporciona información acerca de la localización, la dispersión, el sesgo y las colas del conjunto de datos que se aprecia de una sola mirada. La localización está representada en la línea que corta la caja y representa la mediana (que está dentro de la caja), la dispersión está dada por la altura de la caja, como por la distancia entre los extremos de los bigotes. El sesgo se observa en la desviación que exista entre la línea de la mediana con relación al centro de la caja, y también la relación entre las longitudes de los bigotes. Las colas se pueden apreciar por la longitud de los bigotes con relación a la altura de la caja, y también por las observaciones que se marcan explícitamente.

2. El gráfico de caja contiene información detallada sobre las observaciones de las colas.

3. La gráfica de caja es fácil de calcular y dibujar.

4. Es de fácil explicación al usuario corriente de estadística.

Existen muchas variaciones de este gráfico, las cuales tratan de involucrar otras caractrerísticas de los datos que en un momento dado puedan ser de interés para el investigador, por ejemplo, a veces se utilizan muescas en la caja para comparar la localización de diferentes muestras y ver si la diferencia es significativa desde el punto de vista estadístico. Otros ponen una marquilla para ubicar la media aritmética, otros deforman la caja para obtener más claridad acerca de la distribución, por ejemplo Benjamini, (1988) crea el gráfico "vaso", en el cual se involucran conceptos de estimación de densidades. Zani, Riani y Corbellini (1998) presentan una generalización del gráfico de caja a dos dimensiones.

El boxplot es obtenido mediante la siguiente función:

```
boxplot(x, ...)
boxplot.default(x, ..., range = 1.5, width = NULL,
        varwidth = FALSE, notch = FALSE, names, boxwex = 0.8,
        data = parent.frame(), plot = TRUE,
        border = par("fg"), col = NULL, log = "", pars = NULL,
        horizontal = FALSE, add = FALSE)
boxplot.formula(formula, data = NULL, subset, na.action, ...)
```

Sus argumentos son:

- x, ...: Los datos que van a ser graficados con el boxplot. Puede ser un conjunto de vectores separados, cada uno correspondiendo a un boxplot componente o una lista que contiene a tales vectores. Alternativamente, una especifiacón de la forma "x $\sim$ g" puede ser dada para indicar que las observaciones en el vector "x" van a ser agrupadas de acuerdo a los niveles del factor "g". En este caso, el argumento "data" puede usarse para par valores para la variables en la especificación.'NA's son permitidos en los datos.

- range: Determina la extensión de los "whiskers" de la caja. Si es positivo, "los whiskers" se extienden hasta el dato más extremo, el cual no es más que el "rango" por rango intercuartílico de la caja. Un valor de cero hace que los "whiskers" se extiendan hasta los datos extremos.

- width: Un vector que da los anchos relativos de las cajas.

- varwidth: Si es "TRUE", las cajas son dibujadas con anchos proporcionales a las raíces cuadradas del número de observaciones en los grupos.

- notch: Si es "TRUE", una cuña es dibujada a cada lado de las cajas. Cuando las cuñas de dos gráficos de caja no se traslapan, entonces las medianas son significativamente diferentes a un nivel del 5%.

- names: Un grupo de etiquetas a ser impresas debajo de cada boxplot.

- boxwex: Un factor de escala a ser aplicado a todas las cajas. Cuando sólo hay unos pocos grupos, la apariencia del gráfico puede mejorarse haciendo las cajas más angostas.

- data: “data.frame”, “list”, o “environment” en el cual los nombres de las variables son evaluados cuando “x” es una fórmula.
- plot: Si es “TRUE” (por defecto) entonces se produce el gráfico. De lo contrario, se producen los resúmenes de los boxplots.
- border: Un vector opcional de colores para las líneas de las cajas y debe ser de longitud igual al número de gráficos.
- col: Si se especifica, los colores que contiene serán usados en el cuerpo de las cajas.
- log: Para indicar si las coordenadas “x” o “y” o serán graficadas en escala logarítmica.
- pars: Los parámetros gráficos pueden ser pasados como argumentos para el boxplot.
- horizontal: Si es “FALSE” (por defecto) los boxplots son dibujados verticalmente.
- add: Si es “TRUE” agrega un boxplot al gráfico actual.
- formula: Una fórmula tal como “y $\sim$ x”.
- data: Un data.frame (o lista) del cual las variables en “fórmula” serán tomadas.
- subset: Un vector opcional para especificar un subconjunto de observaciones a ser usadas en el proceso de ajuste.
- na.action: Una función la cual indica lo que debería pasar cuando los datos contienen “NA’s”.

En el siguiente ejemplo, se grafican simultáneamente los datos sobre el número de pasajeros por viaje registrados entre octubre 1 de 1997 a agosto 30 de 1999, de la ruta Aranjuez - Anillo, en donde un viaje constituye el recorrido de salida y regreso de un bus desde la terminal:

```
viajes97<-matrix(scan('a:/pasaje97.txt'),ncol=4,byrow=T)
viajes98<-matrix(scan('a:/pasaje98.txt'),ncol=4,byrow=T)
viajes99<-matrix(scan('a:/pasaje99.txt'),ncol=4,byrow=T)
```

```
pasajeros.viaj97<-matrix((viajes97[,4]/viajes97[,3]),ncol=1)
pasajeros.viaj98<-matrix((viajes98[,4]/viajes98[,3]),ncol=1)
pasajeros.viaj99<-matrix((viajes99[,4]/viajes99[,3]),ncol=1)
anos<-c(rep(97,nrow(viajes97)),rep(98,nrow(viajes98)),
rep(99,nrow(viajes99)))
pasajeros.viaje<-rbind(pasajeros.viaj97,pasajeros.viaj98,
pasajeros.viaj99)
boxplot(split(pasajeros.viaje,anos),
main='Promedio de pasajeros
transportados\nporviajes- Año', xlab='años',
ylab='número por viaje')
#otraforma:

pasajeros97<-cbind(rep(97,nrow(viajes97)),pasajeros.viaj97)
pasajeros98<-cbind(rep(98,nrow(viajes98)),pasajeros.viaj98)
pasajeros99<-cbind(rep(99,nrow(viajes99)),pasajeros.viaj99)
pasajeros<-rbind(pasajeros97,pasajeros98,pasajeros99)
boxplot(pasajeros[,2]~pasajeros[,1],
main='Promedio de pasajeros transportados\npor
viajes - Año', xlab='años',ylab='número por viaje')
```

Una variante del boxplot, es el notched boxplot de McGill, Larsen y Tukey, en el cual el gráfico regular es suplementado con intervalos de confianza para la mediana, representados con un par de cuñas a los lados de la caja, como se observa en la figura 2.2.

```
boxplot(pasajeros[,2]~pasajeros[,1],notch=TRUE,
main='Promedio de pasajeros
transportados\npor viaje - Año', xlab='años',
ylab='número por viaje')
```

Una comparación de los mismos datos por mes, puede realizarse de la siguiente manera:

```
viajes97<-matrix(scan('c:/graficosr/graficos/pasaje97.txt'),
ncol=4,byrow=T)
viajes98<-matrix(scan('c:/graficosr/graficos/pasaje98.txt'),
ncol=4,byrow=T)
```

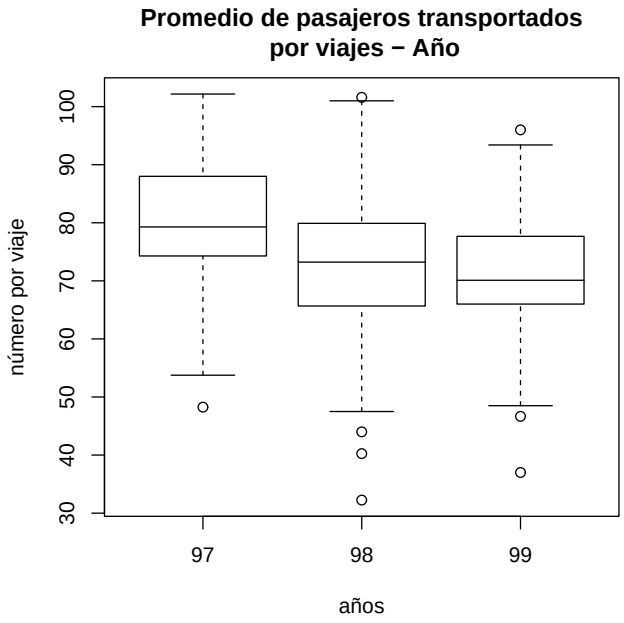

Figura 2.1: *Podemos comparar las distribuciones del número de pasajeros por viaje durante tres años. Se puede apreciar simultáneamente la tendencia y la dispersión.*

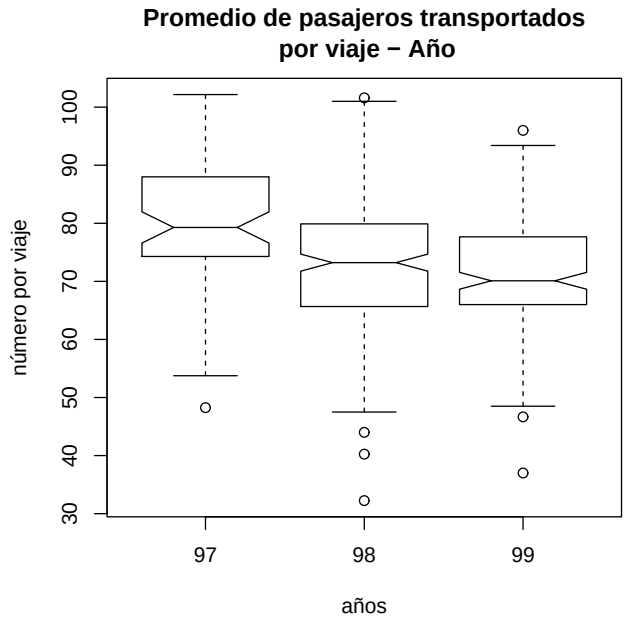

Figura 2.2: *Boxplots con intervalos de confianza para las medianas*

```
viajes99<-matrix(scan('c:/graficosr/graficos/pasaje99.txt'),
ncol=4,byrow=T)
viajes<-matrix(rbind(viajes97,viajes98,viajes99),ncol=4)
boxplot((viajes[,4]/viajes[,3])~viajes[,1],notch=TRUE,
main='Distribuci\'on de Pasajeros
- Viaje por Mes',xlab='mes')
title(sub='Ruta Aranjuez Anillo')
```

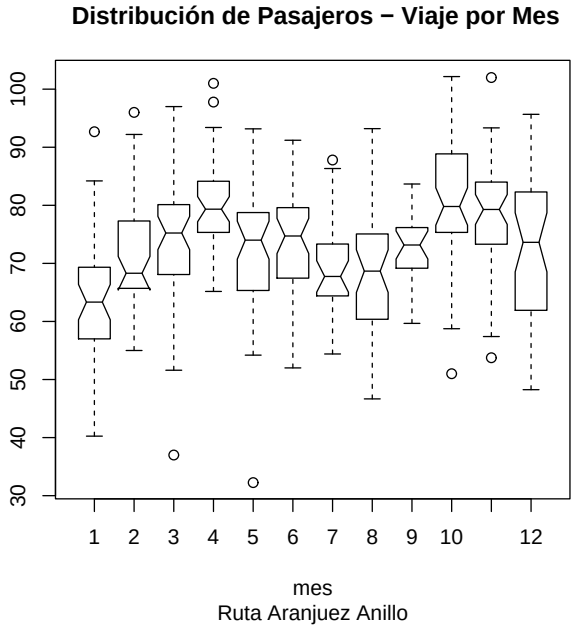

Figura 2.3: *Podemos comparar las distribuciones del número de pasajeros por viaje por mes. Se puede apreciar simultáneamente un comportamiento estacional aproximadamente cada cuatro meses (ascenso, descenso y ascenso), y la dispersión.*

2.2.3 Histogramas

El histograma es el gráfico estadístico por excelencia. El histograma de un conjunto de datos es un gráfico de barras que representan las frecuencias

con que aparecen las mediciones agrupadas en ciertos rangos o intervalos. Para uno construir un histograma se debe dividir la recta real en intervalos o clases (algunos recomiendan que sean de igual longitud) y luego contar cuántas observaciones caen en cada intervalo. Es tal vez el único gráfico que ha tenido un desarrollo teórico en un área que se conoce com estimación de densidades (Scott, 1992). La idea de agrupar datos en forma de histogramas se conoce desde 1662 con el trabajo de Graunt. Sin embargo, es hasta 1926 cuando aparecen las primeras reglas sobre su contrucción con la fórmula de Sturges para determinar el número de barras.

$$\text{Regla de Sturges: } k = 1 + \log_2(n)$$

donde k es el número de barras y n el tamaño muestral.[1]

Scott (1979), considerando el histograma como un estimador de una densidad poblacional,$f(x)$, y basado en una muestra aleatoria, muestra que bajo ciertas condiciones la amplitud de ventana óptima para el histograma es

$$h_n = \left\{ \frac{6}{\int_{-\infty}^{\infty} f'(x)^2 dx} \right\}^{1/3} n^{-1/3}$$

Para una distribución normal el valor óptimo es

$$h_n = 2 \times 3^{1/3} \pi^{1/6} \sigma n^{-1/3}.$$

Scott (1992), basado en la distribución normal recomienda el siguiente número de barras para el histograma

$$\text{Regla de Scott: } k = (2n)^{1/3}$$

Los pasos para construir el histograma son:

1. Defina los intervalos o clases de igual longitud.
2. Cuente el número de observaciones que caen en cada clase o intervalo. Esto es llamado la *frecuencia*.

[1]Esta regla viene de la siguiente igualdad

$$n = \sum_{i=0}^{k-1} \left(\begin{array}{c} k-1 \\ i \end{array} \right) = (1+1)^{k-1} = 2^{k-1}$$

3. Calcule la *frecuencia relativa*,

$$FR = \frac{\text{Nro. de obs. en el intervalo}}{\text{Número de datos, } n}$$

4. Grafique los rectángulos cuyas alturas son proporcionales a las frecuencias relativas.

- Realizar histogramas de esta manera tiene las siguientes ventajas

    1. Es útil para apreciar la forma de la distribución de los datos, si se escoge adecuadamente el número de clases y su amplitud.
    2. Se puede presentar como un gráfico definitivo en un reporte.
    3. Se puede utilizar para comparar dos o más muestras o poblaciones.
    4. Se puede refinar para crear gráficos más especializados, por ejemplo la pirámide poblacional.

- Desventajas

    1. Las observaciones individuales se pierden.
    2. La selección del número de clases y su amplitud que adecuadamente representen la distribución puede ser complicado. Un histograma con muy pocas clases agrupa demasiadas observaciones y uno con muchas deja muy pocas en cada clase. Ninguno de los dos extremos es adecuado.

Debido a que nuestros ojos responden al área de las barras, es importante mantener la anchura de las barras iguales. Si estamos enfrentados a un problema donde los intervalos tienen diferente amplitud, por ejemplo cuando obtenemos datos agrupados desde la fuente, la siguiente fórmula se usa

$$\text{Altura del rectángulo} = \frac{\text{Frecuencia Relativa}}{\text{Amplitud del Intervalo}}$$

Nota: Los programas de computador usualmente ajustan los histogramas automáticamente, pero el programa debe permitirnos variar el histograma. Si usted posee un programa que no le permita hacer cambios, cambie de programa.

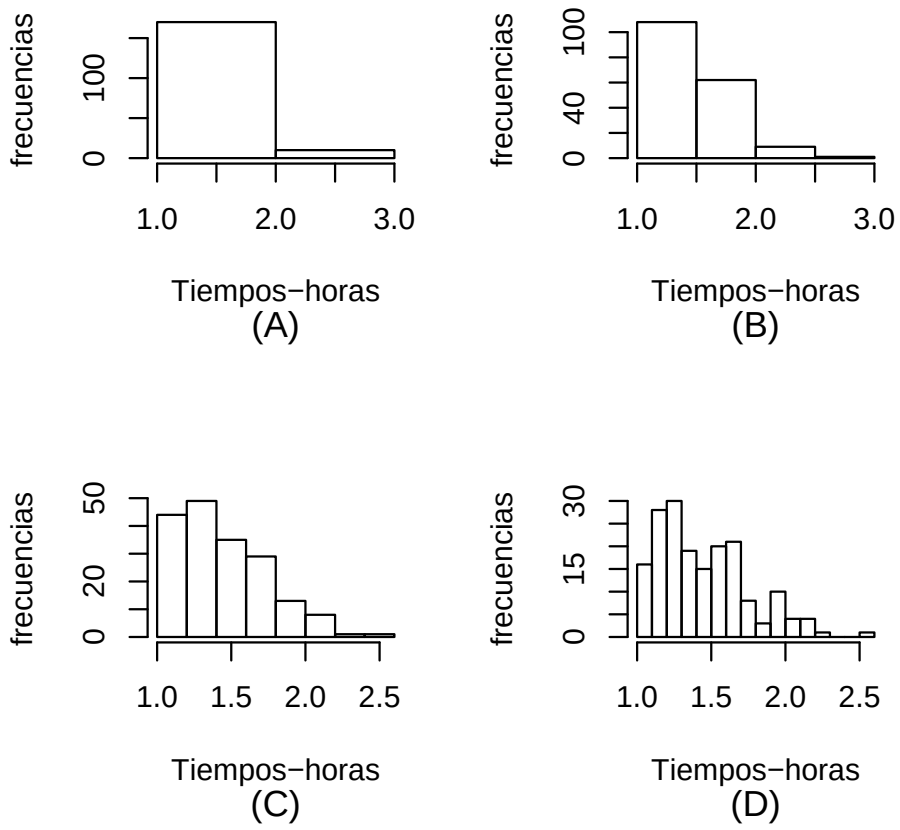

Figura 2.4: *Se muestra la distribución del tiempo utilizado por los atletas masculinos clasificados en el grupo élite en la media maratón de CONAVI. El histograma A tiene solo 2 barras. El gráfico B, con 4 barras,y el C, con 8 barras, muestra más claramente la asimetría (este es el que la mayoría de los programas produce por defecto, ya que la regla de Sturges, para este conjunto de datos aproxima a 8 barras). Si consideramos más barras por ejemplo 16, como tenemos en D, se refina más la información y empezamos a notar multimodalidad.*

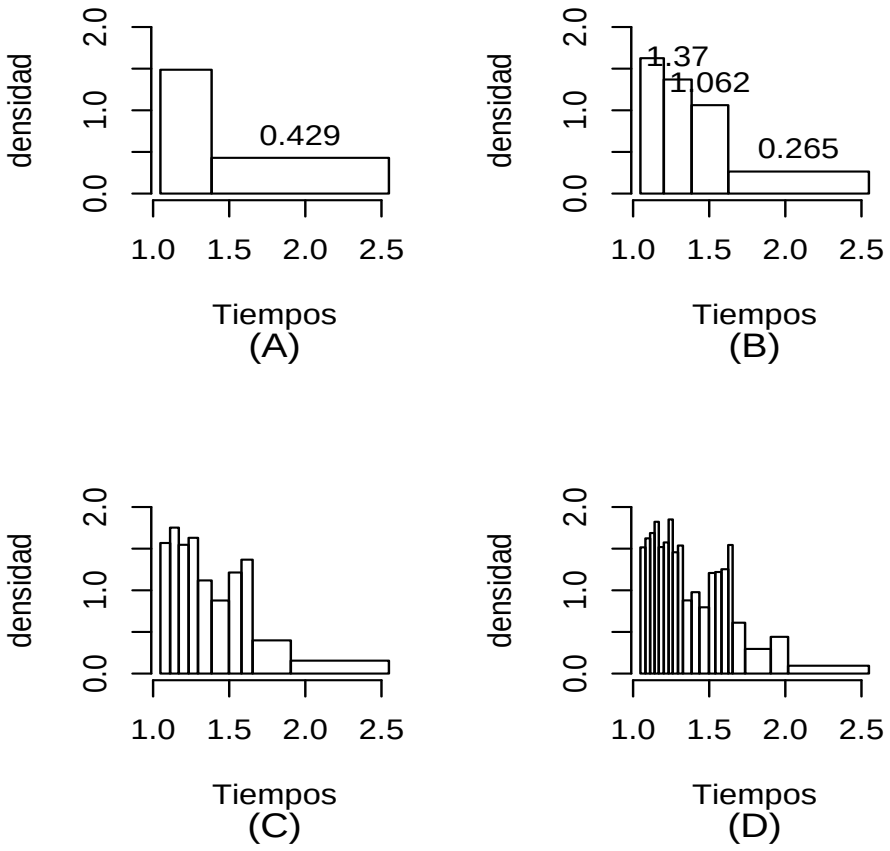

Figura 2.5: *La selección de histogramas con barras de amplitud variable no es recomendada por diversos autores, pero si lo hacemos con cuidado podemos obtener gráficos muy ilustrativos. Se seleccionaron amplitudes de tal forma que cada barra posea el mismo número de observaciones. El histograma A tiene barras que contienen el 50% de las observaciones, el B con 25%, el C con 10% y el D con 5%. La multimodalidad es más clara.*

En R el comando básico con el cual se obtiene este tipo de gráfico es "hist", veamos:

```
hist(x, ...)
     hist.default(x, breaks, freq = NULL, probability = !freq,
          include.lowest = TRUE,
          right = TRUE, col = NULL, border = par("fg"),
          main = paste("Histogram of" , xname),
          xlim = range(breaks), ylim = NULL,
          xlab = xname, ylab,
          axes = TRUE, plot = TRUE, labels = FALSE,
          nclass = NULL, ...)
```

Los argumentos de este comando son:

- x: Vector de valores para el que se construye el histograma.
- breaks: Puede ser un valor con el cual se indica el número aproximado de clases o un vector cuyos elementos indican los puntos límites entre las clases o intervalos.
- freq: argumento lógico; si se especifica como "TRUE", el histograma representará las frecuencias absolutas o conteo de datos en cada clase; si se especifica como "FALSE", se reprentarán las frecuencias relativas. Por defecto, este argumento toma el valor de "TRUE" siempre y cuando los intervalos sean de igual ancho.
- probability: Especifica un alias para "!freq", para compatibilidad con S.
- include.lowest: Argumento lógico; si se especifica como "TRUE", un "x[i]" igual a los equal a un valor "breaks" se incluirá en la primera barra, si el argumento "right = TRUE", o en la última en caso contrario.
- right: Argumento lógico; si es "TRUE", los intervalos son abiertos a la izquierda - cerrados a la derecha (a,b]. Para la primera clase o intervalo si "include.lowest=TRUE" el valor más pequeño de los datos será incluido en éste. Si es "FALSE" los intervalos serán de la forma [a,b) y el argumento "include.lowest=TRUE" tendrá el significado de incluir el "más alto".

- col: Para definir el color de las barras. Poe defecto, “NULL” produce barras sin fondo.
- border: Para definir el color de los bordes de las barras.
- plot: Argumento lógico. Por defecto es “TRUE”, y el resultado es el gráfico del histograma; si se especifica como “FALSE” el resultado es una lista de conteos por cada intervalo.
- labels: Argumento lógico o carácter. Si se especifica como “TRUE” coloca etiquetas arriba de cada barra.
- nclass: Argumento numérico (entero). Para compatibilidad con S, “nclass=n” es equivalente a “breaks=n”.
- ...: Parámetros gráficos adicionales a “title” y “axis”.

```
> velocidades<-c(scan('a:volador.dat'))
Read 52 items
> velocidades

 [1] 60.2 43.3 51.2 46.6 32.5 41.8 45.9 60.6 32.3 31.7 39.4 41.2 60.2 49.0 40.5 58.3 42.7 61.4
[19] 26.0 53.3 58.7 46.4 39.1 63.9 51.5 53.3 41.6 54.9 55.2 60.2 47.3 39.8 46.8 64.4 57.9
39.1 [37] 44.8 65.3 69.7 50.4 54.2 39.4 46.6 55.8 53.6 61.8 44.3 48.5 53.9 61.4 38.1 47.8

> summary(velocidades)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  26.00   41.75   48.75   49.50   58.00   69.70
> stem(velocidades)

  The decimal point is 1 digit(s) to the right of the |

  2 | 6
  3 | 223
  3 | 89999
  4 | 01122334
  4 | 5667777899
  5 | 01233444
  5 | 556889
  6 | 000111244
  6 | 5
  7 | 0
```

```
> hist(velocidades)
```



```
> hist(velocidades,nclass=10)
```

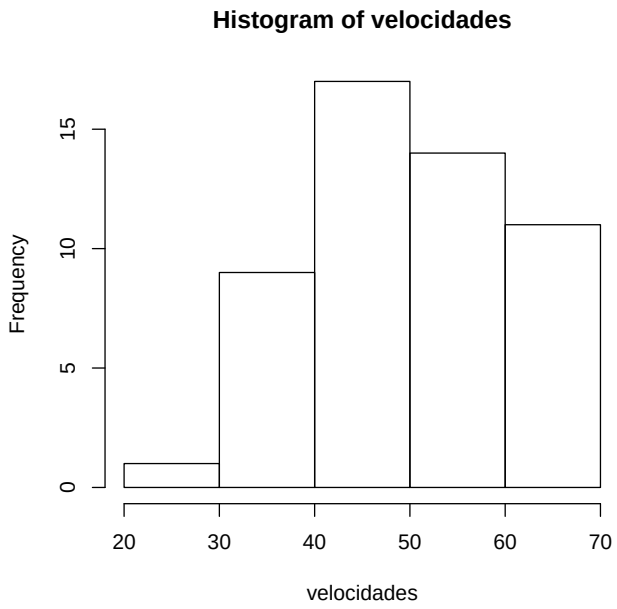

Figura 2.6: *Con la instrucción hist(velocidades), obtenemos un histograma que por defecto R construye con seis (6) intervalos o clases.*

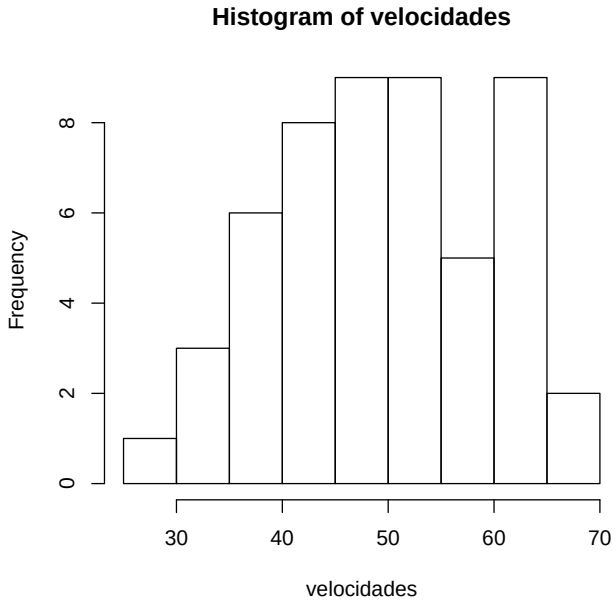

Figura 2.7: *La instrucción nclass=10 modifica el número de clases a 10.*

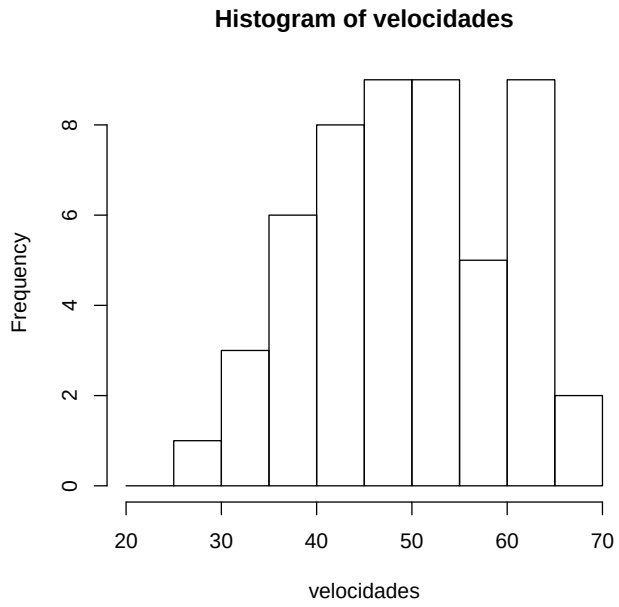

Figura 2.8: *La instrucción breaks=seq(20,70,by=5) proporciona otra forma de obtener un efecto similar al del gráfico anterior especificando los límites deseados para cada clase. Se debe tener cuidad de cubrir todos los valores del rango de la variable.*

```
> hist(velocidades,breaks=seq(20,70,by=5))
```

```
> hist(velocidades,nclass=10,main='Velocidad automoviles
via el Volador',ylab='Frecuencias',xlab='Velocidad en Km/h')
```

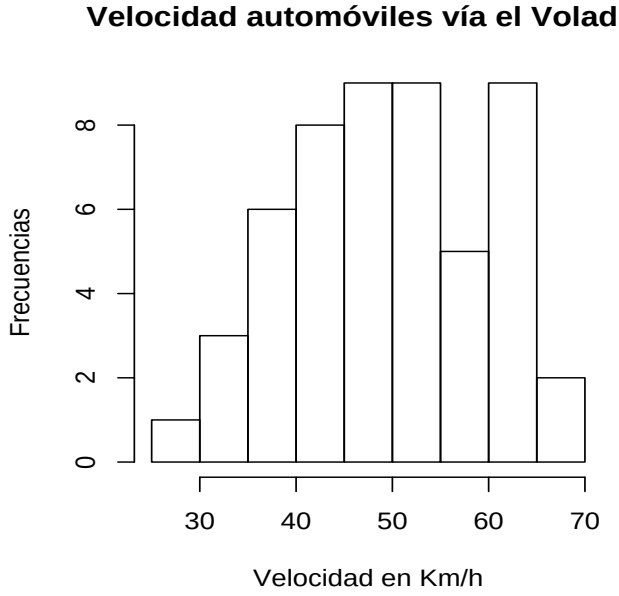

Figura 2.9: *Las instrucciones 'main', 'xlab' y 'ylab' permite modificar el título de gráfico, y las leyendas de ejes "x" y "y" respectivamente.*

```
> hist(velocidades,col='4',nclass=10,
main='Velocidad automoviles
via el Volador',ylab='Frecuencias',xlab='Velocidad en Km/h')
> hist(velocidades,col='blue',nclass=10,
main='Velocidad automoviles
via el Volador',ylab='Frecuencias',xlab='Velocidad en Km/h')
```

Las dos anteriores especificaciones para 'col' tienen el mismo efecto, colorear de azul el fondo de las barras. Los códigos de color, son:

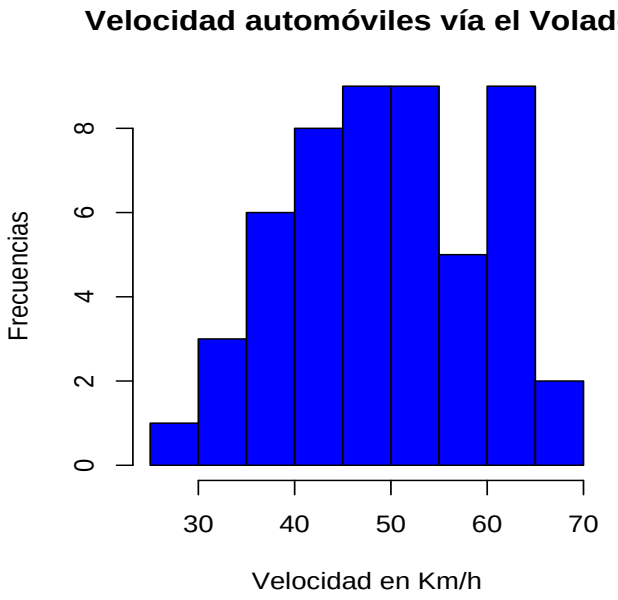

Figura 2.10: *El fondo de las barras puede modificarse con el comando 'col' bien sea mediante un código nuérico o el nombre en inglés del color.*

- 0, 8 blanco
- 1, 9 negro
- 2, 10 rojo
- 3, 11 verde
- 4, 12 azul
- 5, 13 magenta
- 6, 14 violeta
- 7, 15 amarillo

Lo siguiente está relacionado con la figura 2.11.

```
> hist(velocidades,col='14',nclass=10,plot=FALSE)
$breaks (l\'\i mites de clase)
 [1] 25 30 35 40 45 50 55 60 65 70

$counts (frecuencias absolutas)
[1] 1 3 6 8 9 9 5 9 2

$intensities (frecuencias relativas)
[1] 0.003846154 0.011538462 0.023076923 0.030769231
0.034615385 0.034615385 0.019230769
[8] 0.034615385 0.007692308

$mids (marcas de clase o puntos medios)
[1] 27.5 32.5 37.5 42.5 47.5 52.5 57.5 62.5 67.5

> par(mfrow=c(1,2))
> hist(velocidades,col='14',nclass=10,ylim=c(0,10),
labels=TRUE,main= 'Velocidad automóviles',
sub='Via el Volador',
ylab='Frecuencias',xlab='Velocidad en Km/h')
> hist(velocidades,col='14',nclass=10,freq=FALSE,
ylim=c(0,0.04), main='Velocidad automóviles',
sub='Vía el Volador', ylab='Frecuencias
relativas',xlab='Velocidad en Km/h')
```

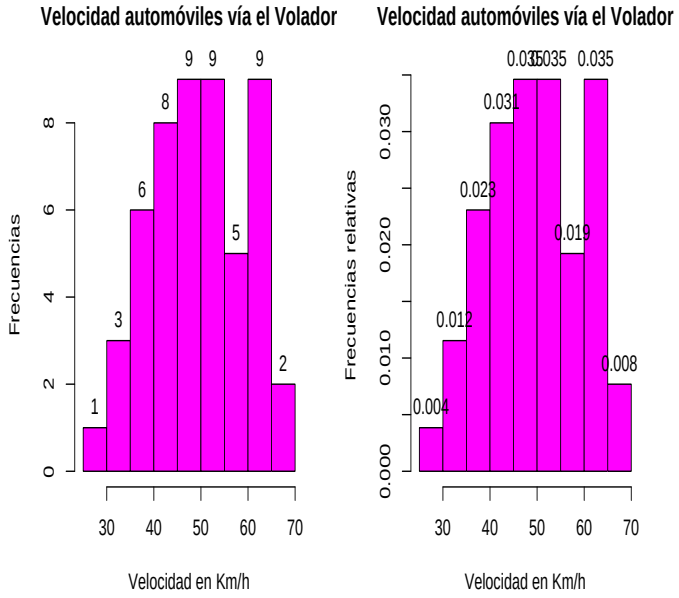

Figura 2.11: *El efecto del argumento freq que por defecto toma el valor de TRUE. Cuando se indica freq=FALSE el segundo histograma presenta la escala vertical en unidades de frecuencia relativa.*

Para los datos de los tiempos de la media maratón de CONAVI, se requirió el siguiente programa:

```
maraton<-scan()

1: 10253 10302 10307 10309 10349 10353 10409 10442 10447 10452 10504 10517
13: 10530 10540 10549 10549 10606 10612 10646 10648 10655 10707 10726 10731
25: 10737 10743 10808 10833 10843 10920 10938 10949 10954 10956 10958 11004
37: 11009 11024 11037 11045 11046 11049 11104 11127 11205 11207 11215 11226
49: 11233 11239 11307 11330 11342 11351 11405 11413 11438 11453 11500 11501
61: 11502 11503 11527 11544 11549 11559 11612 11617 11635 11655 11731 11735
73: 11746 11800 11814 11828 11832 11841 11909 11926 11937 11940 11947 11952
85: 12005 12044 12113 12209 12230 12258 12309 12327 12341 12413 12433 12440
97: 12447 12530 12600 12617 12640 12700 12706 12727 12840 12851 12851 12937
109: 13019 13040 13110 13114 13122 13155 13205 13210 13220 13228 13307 13316
121: 13335 13420 13425 13435 13435 13448 13456 13536 13608 13612 13620 13646
133: 13705 13730 13730 13730 13747 13810 13850 13854 13901 13905 13907 13912
145: 13920 14000 14010 14025 14152 14208 14230 14344 14400 14455 14509 14552
157: 14652 15009 15026 15242 15406 15409 15528 15549 15644 15758 15837 15916
169: 15926 15948 20055 20416 20520 20600 20732 20748 20916 21149 21714 23256
181:

Read 180 items
horas<-maraton%/%10000
min<-(maraton-horas*10000)%/%100
seg<-maraton-horas*10000-min*100
Tiempos<-horas+min/60+seg/3600
> Tiempos

  [1] 1.048056 1.050556 1.051944 1.052500 1.063611 1.064722 1.069167 1.078333
  [9] 1.079722 1.081111 1.084444 1.088056 1.091667 1.094444 1.096944 1.096944
 [17] 1.101667 1.103333 1.112778 1.113333 1.115278 1.118611 1.123889 1.125278
 [25] 1.126944 1.128611 1.135556 1.142500 1.145278 1.155556 1.160556 1.163611
 [33] 1.165000 1.165556 1.166111 1.167778 1.169167 1.173333 1.176944 1.179167
 [41] 1.179444 1.180278 1.184444 1.190833 1.201389 1.201944 1.204167 1.207222
 [49] 1.209167 1.210833 1.218611 1.225000 1.228333 1.230833 1.234722 1.236944
 [57] 1.243889 1.248056 1.250000 1.250278 1.250556 1.250833 1.257500 1.262222
 [65] 1.263611 1.266389 1.270000 1.271389 1.276389 1.281944 1.291944 1.293056
 [73] 1.296111 1.300000 1.303889 1.307778 1.308889 1.311389 1.319167 1.323889
 [81] 1.326944 1.327778 1.329722 1.331111 1.334722 1.345556 1.353611 1.369167
 [89] 1.375000 1.382778 1.385833 1.390833 1.394722 1.403611 1.409167 1.411111
 [97] 1.413056 1.425000 1.433333 1.438056 1.444444 1.450000 1.451667 1.457500
[105] 1.477778 1.480833 1.480833 1.493611 1.505278 1.511111 1.519444 1.520556
[113] 1.522778 1.531944 1.534722 1.536111 1.538889 1.541111 1.551944 1.554444
[121] 1.559722 1.572222 1.573611 1.576389 1.576389 1.580000 1.582222 1.593333
[129] 1.602222 1.603333 1.605556 1.612778 1.618056 1.625000 1.625000 1.625000
[137] 1.629722 1.636111 1.647222 1.648333 1.650278 1.651389 1.651944 1.653333
[145] 1.655556 1.666667 1.669444 1.673611 1.697778 1.702222 1.708333 1.728889
[153] 1.733333 1.748611 1.752500 1.764444 1.781111 1.835833 1.840556 1.878333
[161] 1.901667 1.902500 1.924444 1.930278 1.945556 1.966111 1.976944 1.987778
[169] 1.990556 1.996667 2.015278 2.071111 2.088889 2.100000 2.125556 2.130000
[177] 2.154444 2.196944 2.287222 2.548889
```

```
par(mfrow=c(2,2)) hist(Tiempos,nclass=2,
main="",xlab='Tiempos-horas',ylab='frecuencias')
mtext("(A)",side=1,line=4,font=1)
hist(Tiempos,nclass=4,main="",xlab='Tiempos-horas',
ylab='frecuencias')
mtext("(B)",side=1,line=4,font=1)
hist(Tiempos,nclass=8,main="",xlab='Tiempos-horas',
ylab='frecuencias')
mtext("(C)",side=1,line=4,font=1)
hist(Tiempos,nclass=16,main="",xlab='Tiempos-horas',
ylab='frecuencias')
mtext("(D)",side=1,line=4,font=1)
puntos<-c(quantile(Tiempos,probs=c(0,0.5,1)))
puntos2<-c(quantile(Tiempos,probs=c(0,0.25,0.5,0.75,1)))
puntos3<-c(quantile(Tiempos,probs=seq(0,1,by=0.10)))
puntos4<-c(quantile(Tiempos,probs=seq(0,1,by=0.05)))
puntos
```

```
      0%      50%     100%
1.048056 1.384306 2.548889
puntos2
      0%      25%      50%      75%     100%
1.048056 1.201806 1.384306 1.625000 2.548889
puntos3
      0%      10%      20%      30%      40%
1.048056 1.111833 1.168889 1.233556 1.294889
     50%      60%      70%      80%      90%
1.384306 1.498278 1.580667 1.653778 1.904694
    100%
2.548889
puntos4
      0%       5%      10%      15%      20%      25%      30%      35%
1.048056 1.081042 1.111833 1.141458 1.168889 1.201806 1.233556 1.260569
     40%      45%      50%      55%      60%      65%      70%      75%
1.294889 1.327403 1.384306 1.435458 1.498278 1.539667 1.580667 1.625000
     80%      85%      90%      95%     100%
1.653778 1.735625 1.904694 2.018069 2.548889
```

```
hist(Tiempos,breaks=puntos,freq=FALSE,ylim=c(0,2),
labels=TRUE,main="",ylab="densidad")
mtext("(A)",side=1,line=4,font=1)
hist(Tiempos,breaks=puntos2,freq=FALSE,ylim=c(0,2),
labels=TRUE,main="",ylab="densidad")
mtext("(B)",side=1,line=4,font=1)
hist(Tiempos,breaks=puntos3,freq=FALSE,ylim=c(0,2),
```

```
main="",ylab="densidad")
mtext("(C)",side=1,line=4,font=1)
hist(Tiempos,breaks=puntos4,freq=FALSE,ylim=c(0,2),
main="",ylab="densidad")
mtext("(D)",side=1,line=4,font=1)
```

Nota:En estos histogramas, las alturas corresponden a las intensidades (frec. relativa/long. intervalo). Por tanto, el área de cada rectángulo da cuenta de las frecuencias relativas. Para el caso (A) ambos intervalos tienen igual área y cada uno contiene 50 % de los datos. esto puede verificarse así:

```
Intensidad primera clase=1.4869888=0.5/(1.384306-1.048056)
Intensidad segunda clase=0.4293381=0.5/(2.548889-1.384306)
```

2.2.4 Gráficos de Puntos

Los gráficos de puntos son elegantemente simples y permite numerosas variaciones. La única razón por la cual no se han vuelto populares es que los programas de hojas electrónicas no los elaboren presionando una tecla (Carr y Sun, 1999).

En R se puede obtener dicho gráfico mediante la función "stripchart":

```
stripchart(x, method="overplot", jitter=0.1, offset=1/3,
  vertical=FALSE, group.names,
  xlim=NULL, ylim=NULL, main="", ylab="", xlab="",
  pch=0, col=par("fg"), cex=par("cex"))
```

Los argumentos de esta función son como sigue:

- x: Objeto que contiene los datos a ser graficados. Puede ser un sólo vector o una lista de vectores, cada uno correspondiendo a un componente del gráfico. También se puede especificar "x g" para indicar que las observaciones en el vector "x" van a ser agrupadas según los niveles del factor "g".

- method: Para especificar el método a ser empleado para separar puntos coincidentes. Si se especifica como "overplot" (por defecto) los puntos coincidentes son superpuestos. Si se especifica como "jitter", para "to

jitter the points", y si se especifica como "stack" los puntos coincidentes son apilados,

- jitter: Para especificar la cantidad de "jitter" a ser aplicado.
- offset: when stacking is used, points are stacked this many line-heights (symbol widths) apart.
- vertical: Argumento lógico. Por defecto es "FALSE" y los puntos se grafican horizontalmente.
- group.names: Grupo de etiquetas a ser impresas al lado o debajo de cada gráfico
- ...: Parámetros gráficos adicionales que pueden especificarse como argumentos.

Veamos el siguiente ejemplo (ver figura 2.12), con base en los tiempos de la media maratón CONAVI:

```
par(mfrow=c(2,2))
stripchart(Tiempos,method='overplot',pch=1,main='Grafico de
puntos',xlab='Tiempos')
mtext("(A)",side=1,line=4,font=1)
stripchart(Tiempos,method='stack',pch=2, main='Grafico de
puntos',xlab='Tiempos')
mtext("(B)",side=1,line=4,font=1)
stripchart(Tiempos,method='jitter',pch=3, main='Grafico
de puntos',xlab='Tiempos')
mtext("(C)",side=1,line=4,font=1)
stripchart(Tiempos,method='stack',vertical=TRUE,pch=0,
main='Grafico de
puntos',ylab='Tiempos')
mtext("(D)",side=1,line=4,font=1)
```

Veamos ahora varios gráficos de puntos para los datos relativos a la composición profesoral y número de estudiantes en 10 universidades de Antioquia, con base en los datos del Anuario Estadístico de Antioquia 1994, obtenidos con el siguiente programa en R:

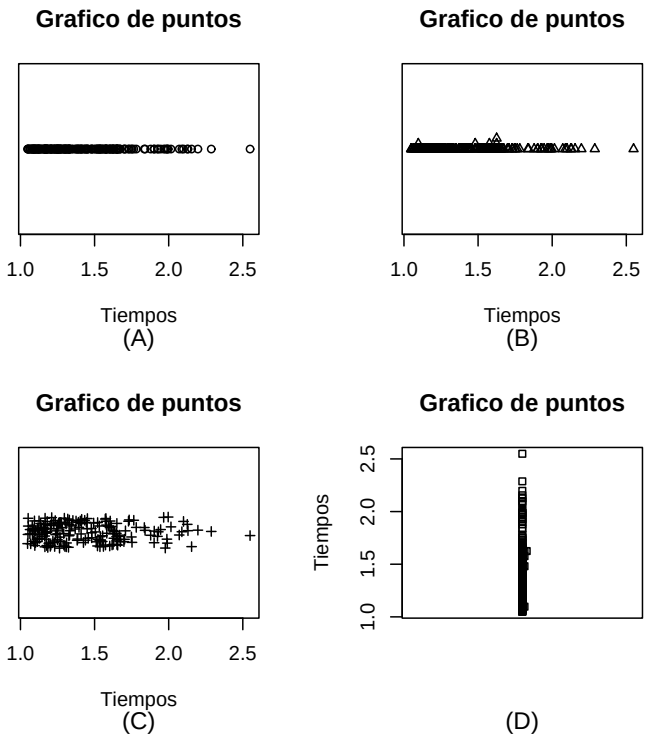

Figura 2.12: *Gráficos de puntos, Tiempos de la media maratón CONAVI.: En A con el método overplot, en B con el método stack, en C con el método jitter y en D presentación vertical con el método stack. Observe los efectos en los símbolos empleados de acuerdo a los valores dados al argumento pch.*

```
datos.prof<-read.table("c:/pili/graficos R/universidades.txt",
header=T)

datos.prof

   Universidad ProfTC ProfMT ProfC Magister PhD Estudiantes
1         UdeA    977    308  1235      454  67       34406
2         UNal    968     82   262      373  79        9495
3        EAFIT    224     17   651       NA  NA        9684
4       EscIng      9      7   182       28   0         688
5          CES     36    227    27        6   1        1582
6       UNAULA      0      0   193       19   1        2006
7          UCC      0      0  1041       26   0        8295
8         UdeM     24     15   786       99  10        5907
9          UPB    111    136   956       98  21       14083
10      USanBu     26     38   182       45   6        3239


stripchart(datos.prof[,2]~datos.prof[,1],pch=19)

stripchart(datos.prof[,2:4],method="overplot",
group.names=colnames(datos.prof[,2:4]),pch=19)

stripchart(datos.prof[,2:7],method="overplot",
group.names=colnames(datos.prof[,2:7]),pch=19)
```

2.2.5 Gráficos Circulares (Pie Charts)

El uso de gráficos circulares o pasteles es bastante común entre personas no profesionales en estadística y lamentablemente se ha trivializado tanto que si en muchas de las situaciones donde se usan se suprimieran se ahorrarían muchas hojas de papel. A veces se presenta un gráfico de pastel para mostrar que en una muestra el 50% son hombres y el 50% mujeres.

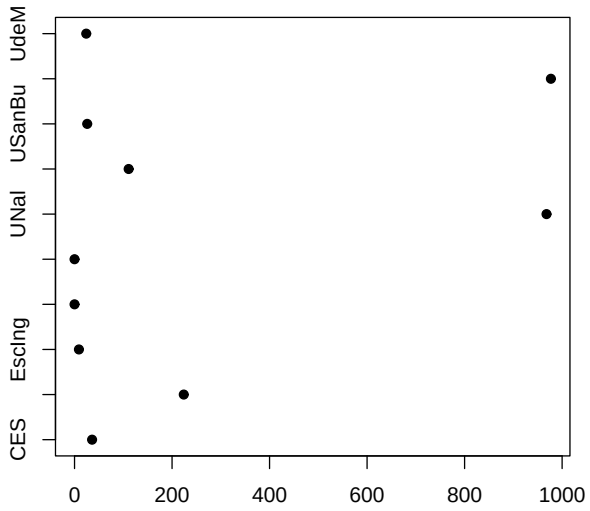

Figura 2.13: *El gráfico de puntos es excelente para realizar comparaciones ya que es muy limpio, esto es, no emplea más tinta de la estrictamente necesaria y la comparación se realiza verticalmente. Tenemos en esta figura el número de profesores de tiempo completo que cada universidad en Antioquia posee (Anuario Estadístico de Antioquia, 1994).*

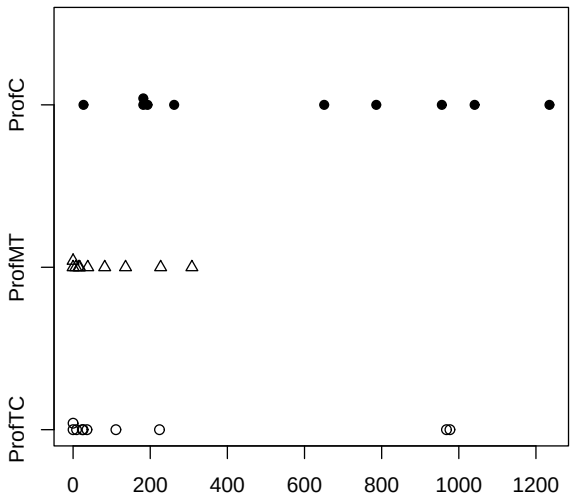

Figura 2.14: *profTC: Tiempo completo, profMT: Medio tiempo y profC: Cátedra. El gráfico de puntos nos permite presentar información agrupada por otra variable de agrupación. Una forma tradicional es realizar un gráfico de torta para cada universidad, lo cual genera 10 gráficos difícilmente comparables(Anuario Estadístico de Antioquia, 1994).*

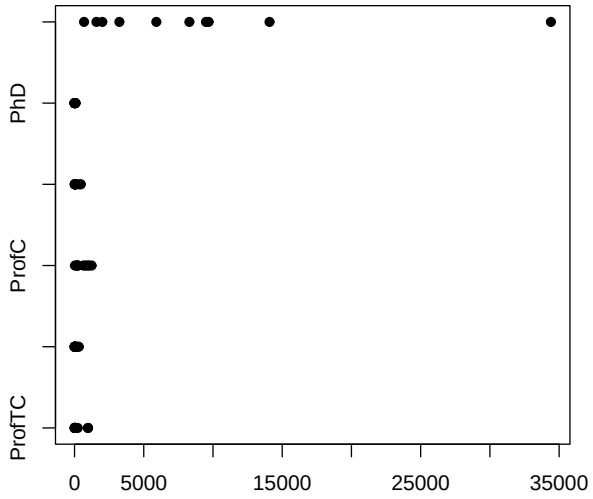

Figura 2.15: *Nuevamente, se presenta la información agrupada de las 6 variables ProfTC ProfMT ProfC Magister PhD Estudiantes, de la base de datos sobre la composición profesoral y el número de estudiantes de 10 de las principales universidades de Antioquia (Anuario Estadístico de Antioquia, 1994).*

Este gráfico es una gran herramienta para datos porcentuales tomadas sobre individuos o elementos, por ejemplo un análisis químico sobre el porcentaje de componentes de muestras tomadas en diversas áreas. En este caso realmente este es un gráfico multivariable que puede utilizarse para comparar diferencias o similitudes y realizar agrupamientos.

En R este tipo de gráfico se obtiene con la función "piechart", veamos:

```
piechart(x, labels=names(x), shadow=FALSE,
         edges=200, radius=0.8, col=NULL, main=NULL, ...)
```

Argumentos:

- x: Vector de cantidades positivas, los cuales son presentados como las áreas en el gráfico.
- labels: Un vector de caracteres "strings" que dan nombres a las áreas.
- shadow: Un vector lógico que indica si un efecto de sombrea será aplicado para el gráfico, cuando se utilizan colores de relleno.
- edges: Aproxima la linea exterior circular mediante un polígono con el número de lados especificado, que por defecto es 200.
- radius: La torta es dibujada centrada en una caja cuadrada cuyos lados se mueven de -1 a 1. Si se usan etiquetas largas puede ser necesario usar radios más pequeños.
- col: Un vector de colores, para rellenar los sectores del gráfico.
- main: Para dar título al gráfico.

Veamos los siguientes ejemplos:

```
datos.antioquia<-scan()
.
.
datos.antioquia<-matrix(datos.antioquia,ncol=15,byrow=T)
```

```
temperatura<-c(datos.antioquia[,2])

tipo<-rep(NA,length(temperatura))
tipo[temperatura<=15]<-'Muy baja'
tipo[15<temperatura & temperatura<=20]<-'Baja'
tipo[20<temperatura & temperatura<=25]<-'Media'
tipo[25<temperatura]<-'Alta'
> tipo
```

```
  [1] "Media"    "Media"    "Media"    "Baja"     "Media"    "Media"
  [7] "Media"    "Media"    "Baja"     "Baja"     "Alta"     "Alta"
 [13] "Alta"     "Alta"     "Alta"     "Alta"     "Media"    "Alta"
 [19] "Alta"     "Alta"     "Media"    "Media"    "Media"    "Media"
 [25] "Media"    "Baja"     "Media"    "Media"    "Media"    "Media"
 [31] "Media"    "Muy baja" "Media"    "Baja"     "Baja"     "Baja"
 [37] "Baja"     "Baja"     "Media"    "Media"    "Muy baja" "Muy baja"
 [43] "Muy baja" "Baja"     "Media"    "Muy baja" "Baja"     "Baja"
 [49] "Media"    "Media"    "Media"    "Media"    "Media"    "Alta"
 [55] "Media"    "Media"    "Baja"     "Baja"     "Baja"     "Baja"
 [61] "Media"    "Baja"     "Baja"     "Baja"     "Baja"     "Baja"
 [67] "Baja"     "Baja"     "Baja"     "Muy baja" "Baja"     "Baja"
 [73] "Baja"     "Media"    "Media"    "Baja"     "Muy baja" "Media"
 [79] "Media"    "Baja"     "Baja"     "Baja"     "Media"    "Baja"
 [85] "Media"    "Baja"     "Baja"     "Muy baja" "Baja"     "Media"
 [91] "Baja"     "Media"    "Baja"     "Media"    "Media"    "Alta"
 [97] "Alta"     "Alta"     "Alta"     "Alta"     "Alta"     "Alta"
```

```
frec.tipo.temperatura<-table(tipo)
piechart(frec.tipo.temperatura, col=rainbow(24))
par(oma=c(1,1,1,1),new=T,font=2,cex=1)
mtext(outer=T,'GRAFICO DE SECTORES PARA TIPO
TEMPERATURA',side=3)
```

Con base en los datos sobre composición profesoral de 10 universidades de Antioquia presentados en secciones previas, se construyen a continuación 10 gráficos de torta para comparar dichas universidades, utilizando el siguiente programa en R:

```
datos.prof1<-data.matrix(datos.prof[,2:4])
datos.prof1
   ProfTC ProfMT ProfC
1     977    308  1235
2     968     82   262
3     224     17   651
```

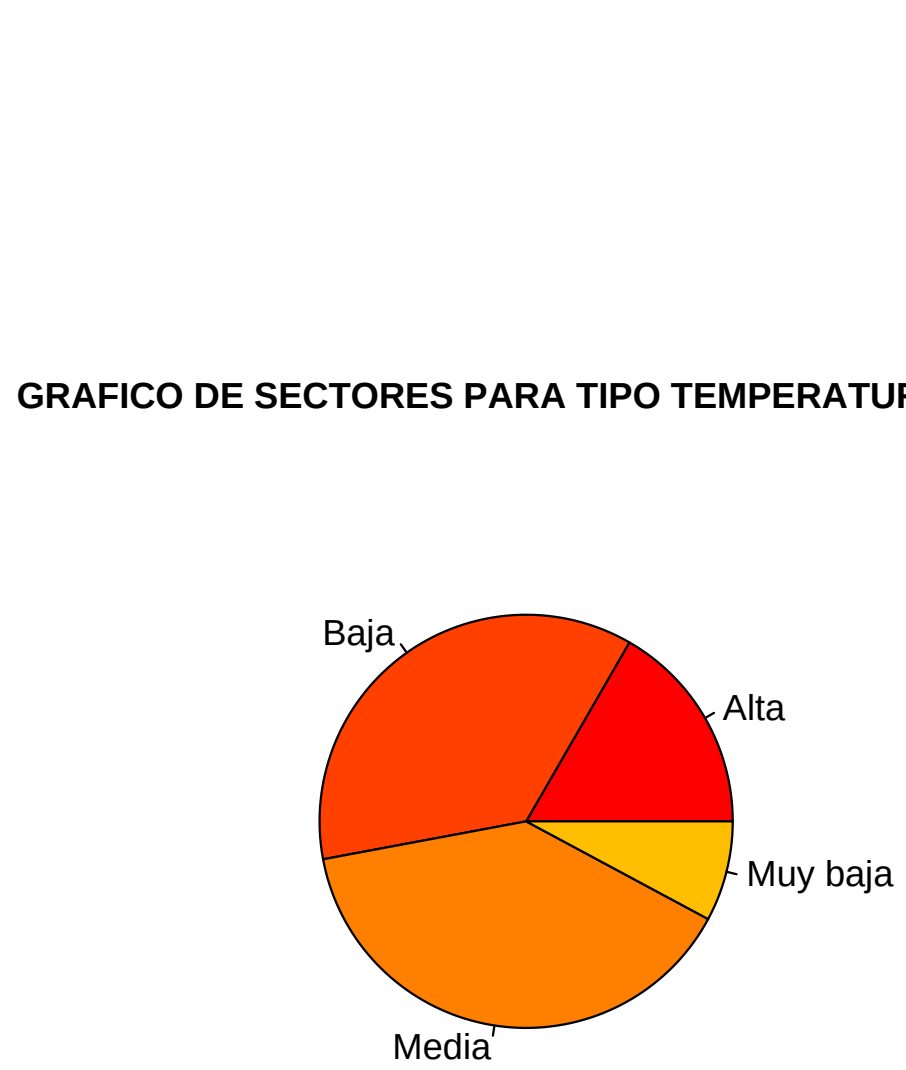

Figura 2.16: *Gráfico de torta para datos de temperaturas de los municipios de Antioquia, por categorías.*

```
4       9      7   182
5      36    227    27
6       0      0   193
7       0      0  1041
8      24     15   786
9     111    136   956
10     26     38   182
universidades<-datos.prof[,1]
> universidades

 [1] UdeA   UNal   EAFIT  EscIng CES    UNAULA UCC
 UdeM   UPB    USanBu
Levels:  CES EAFIT EscIng UCC UNAULA UNal UPB USanBu UdeA UdeM

par(mfrow=c(4,3))
for(i in 1:10){
datos<-datos.prof1[i,][datos.prof1[i,]>0]

#Para garantizar que los valores a
#graficar sean no nulos, es decir, mayores que cero

piechart(datos,main=universidades[i]) }
```

2.2.6 Gráfico de barras

Este gráfico es una modificación más clara del gráfico de torta.

Los gráficos de barras son suficientemente flexibles para ser adaptados a situaciones donde el trabajo gráfico ha tenido poco éxito, como lo es el análisis de datos categóricos. Hofmann (1998-1999) analizó el problema de la Paradoja de Simpson (un problema que surge cuando se trabajan con tablas de contingencia marginales en lugar de una tabla completa) en el caso de la tragedia del Titanic, utilizando solo gráficos de barras.

Allen y Buckner (1992) presentan variaciones de los gráficos de barras en las cuales muestran la gran variedad de alternativas que surgen a partir de este sencillo gráfico, en especial cuando se mezclan con otros gráficos, por

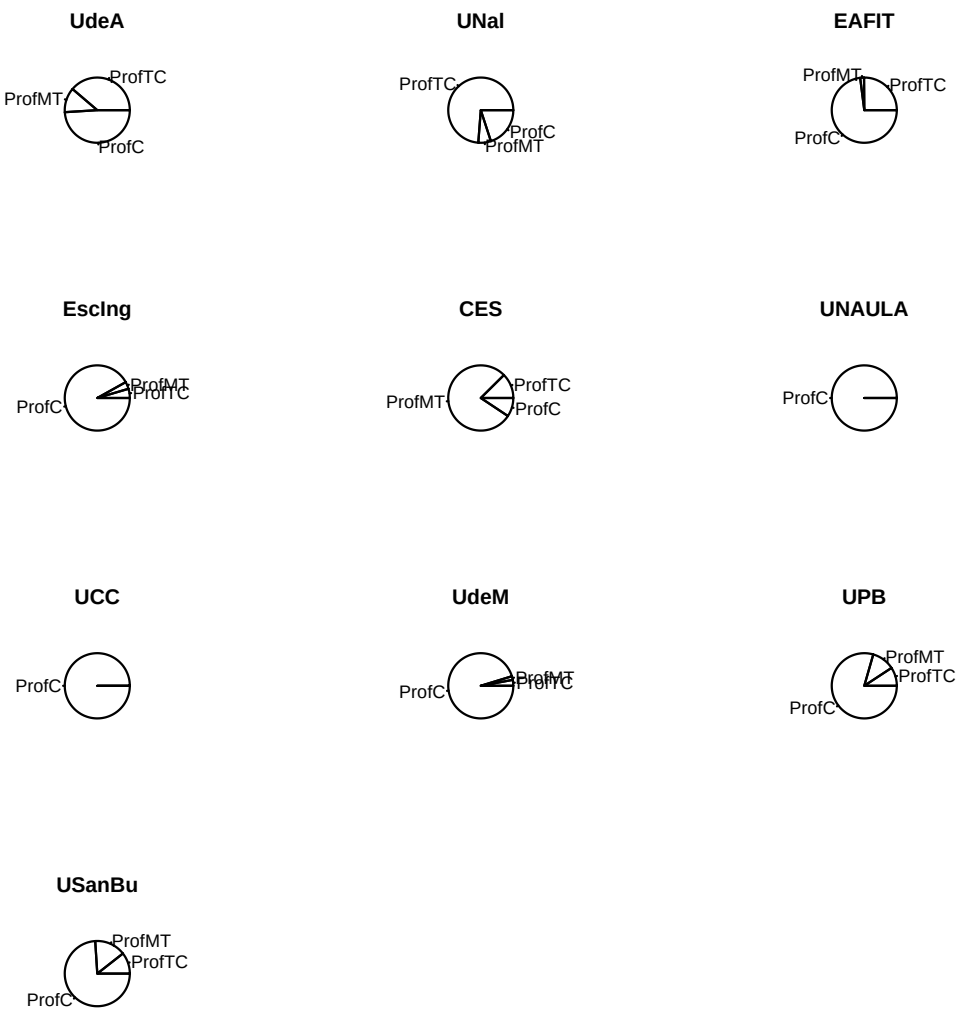

Figura 2.17: *Un gráfico de torta para cada universidad nos genera 10 gráficos que presenta información obligatoriamente porcentual, razón por la cual la Universidad de Antioquia, a pesar de tener casi igual número de profesores de tiempo completo que la Universidad Nacional, se ven tan diferentes (Anuario Estadístico de Antioquia, 1994).*

ejemplo el de torta, histogramas o con información estadística, como lo es el error estándar.

Para obtener este tipo de gráficos en R, la función base es "barplot", como se describe a continuación:

```
barplot(height, width = 1, space = NULL, names.arg = NULL,
legend.text = NULL, beside = FALSE, horiz = FALSE,
col = heat.colors(NR), border = par("fg"),
main = NULL, sub = NULL, xlab = NULL, ylab = NULL,
xlim = NULL, ylim = NULL,
axes = TRUE, axisnames = TRUE, inside = TRUE, plot = TRUE, ...)
```

Los argumentos son como sigue:

- height: Vector o matriz de valores que describen las barras. Si es un vector, entonces el gráfico corresponde a un secuencia de barras rectangulares cuyas alturas corresponden a los valores del vector. Si es una matriz y "beside=FALSE" entonces cada barra del gráfico corresponde a una columna de forma que los valores en cada columna serán representados por sub barras apiladas, pero si "beside=TRUE" entonces los valores en cada columna serán yuxtapuestos en vez de apilados.

- width: Es opcional, para especificar mediante un vector el ancho de las barras.

- space: Para fijar el espacio entre barras, como una fracción del ancho promedio de éstas. Puede especificarse con un sólo número o un número por barra. En el caso de que el objeto representado sea una matriz y "beside=TRUE", este argumento puede darse por dos números el primero para el espacio entre barras del mismo grupo y el segundo para el espacio entre grupos. Por defecto está ajustado a 'c(0,1)' para el caso de objeto matriz y beside=TRUE.

- names.arg: Un vector de nombres para colocarlos debajo de cada barra o grupo de barras. Si se omite, entonces los nombres son tomados de los atributos de nombres que estéén contenidos en el objeto especificado en "height".

- legend.text: Un vector de texto para construir una leyenda para el gráfico. Sólo es útil si "height" es una matriz, en cuyo caso las leyendas corresponderán a sus filas.

- beside: Un valor lógico. Si es “FALSE” las columnas de “height” serán representadas por barras apiladas y si es “TRUE” entonces serán representadas por barras yuxtapuestas.

- horiz: Un valor lógico. Si es “FALSE” las barras son dibujadas verticalmente.

- col: Para especificar un vector de colores para las barras.

- border: Para indicar el color a ser usado en los bordes de las barras.

- main,sub: Para titular y subtitular el gráfico.

- xlab: Para dar nombre al eje x.

- ylab: Para dar nombre al eje y.

- xlim: Para delimitar el rango de valores en el eje x.

- ylim: Para delimitar el rango de valores en el eje y.

- axes: Argumento lógico. Si es “TRUE”, dibuja el correspondiente eje.

- axisnames: Argumento lógico. Si es “TRUE”, y si se ha especificado “names.arg”, entonces el otro eje es dibujado y etiquetado.

- plot: Argumento lógico. Si es “FALSE” no se produce gráfico

Nuevamente, para los datos de temperatura de los municipios de Antioquia, veamos el correspondiente gráfico de barras:

```
barplot(frec.tipo.temperatura,space=0,
col=c('gray30','gray40','gray50','gray60'),
main='Temperaturas en Antioquia',ylab='Frecuencias')
```

Otro ejemplo, cruzando los datos de población y de temperaturas, se puede obtener una tabla de frecuencias de la distribución de la población en los municipios de Antioquia, de acuerdo a los niveles de temperatura, la cual puede ser representada con un diagrama de barras, así :

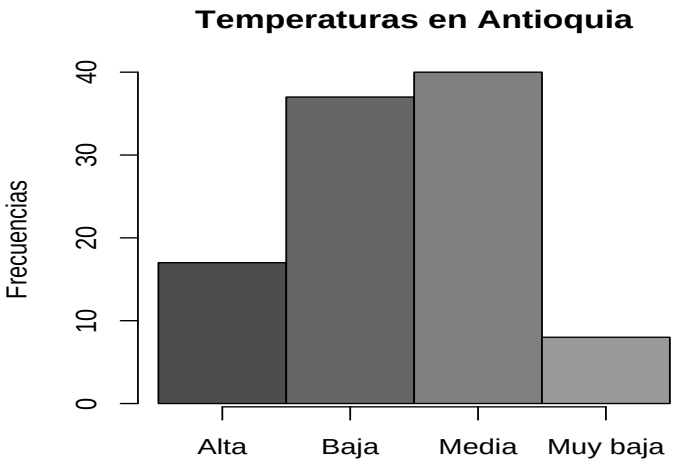

Figura 2.18: *Las frecuencias por categoría son Alta: 17 Baja:37 Media:40 Muy baja:8*

```
poblacion<-c(datos.antioquia[,3])
> tipo.poblacion<-rep(NA,length(poblacion))
> summary(poblacion)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2959   10350   17070   44720   30040 1835000
> tipo.poblacion[2500<poblacion & poblacion<=10000]<-'Muy Baja'
> tipo.poblacion[10000<poblacion & poblacion<=30000]<-'Baja'
> tipo.poblacion[30000<poblacion & poblacion<=100000]<-'Media'
> tipo.poblacion[100000<poblacion & poblacion<=500000]<-'Alta'
> tipo.poblacion[500000<poblacion]<-'Muy Alta'
> tipo.poblacion

  [1] "Muy Alta" "Media"    "Alta"     "Media"    "Media"    "Alta"
  [7] "Media"    "Alta"     "Media"    "Baja"     "Baja"     "Media"
 [13] "Baja"     "Baja"     "Baja"     "Muy Baja" "Muy Baja" "Media"
 [19] "Baja"     "Baja"     "Baja"     "Baja"     "Baja"     "Baja"
 [25] "Baja"     "Baja"     "Media"    "Baja"     "Muy Baja" "Baja"
 [31] "Baja"     "Muy Baja" "Muy Baja" "Baja"     "Muy Baja" "Baja"
 [37] "Muy Baja" "Muy Baja" "Media"    "Muy Baja" "Muy Baja" "Baja"
 [43] "Baja"     "Muy Baja" "Baja"     "Media"    "Muy Baja" "Muy Baja"
 [49] "Baja"     "Baja"     "Baja"     "Muy Baja" "Baja"     "Baja"
```

```
 [55] "Baja"     "Muy Baja" "Baja"     "Muy Baja" "Baja"     "Media"
 [61] "Baja"     "Muy Baja" "Baja"     "Baja"     "Baja"     "Baja"
 [67] "Baja"     "Muy Baja" "Media"    "Baja"     "Media"    "Baja"
 [73] "Media"    "Baja"     "Baja"     "Baja"     "Media"    "Baja"
 [79] "Media"    "Muy Baja" "Baja"     "Muy Baja" "Media"    "Baja"
 [85] "Muy Baja" "Baja"     "Baja"     "Muy Baja" "Baja"     "Baja"
 [91] "Media"    "Baja"     "Media"    "Muy Baja" "Baja"     "Media"
 [97] "Baja"     "Media"    "Muy Baja" "Baja"     "Media"    "Muy Baja"
```

```
frec.tipos.temperatura.poblacion<-table(tipo.poblacion,
tipo.temperatura)
frec.tipos.temperatura.poblacion
              tipo.temperatura
tipo.poblacion Alta Baja Media Muy baja
      Alta        0    0     3        0
      Baja        9   18    21        3
      Media       5    8     7        2
      Muy Alta    0    0     1        0
      Muy Baja    3   11     8        3
```

```
barplot(frec.tipos.temperatura.poblacion,beside=T,
legend.text=
rownames(frec.tipos.temperatura.poblacion), ylim=c(-0.5,25),
main='Distribuci\'on de
poblaci\'on por rangos de temperatura\nen Antioquia',
xlab='Tipo Temperatura',ylab='frec.
por tipo de pob')
barplot(frec.tipos.temperatura.poblacion,
beside=FALSE,legend.text=
rownames(frec.tipos.temperatura.poblacion),
main='Distribuci\'on de poblaci\'on por
rangos de temperatura\nen Antioquia', xlab='Tipo Temperatura',
ylab='frec.por tipo de
pob')
```

Los gráficos resultantes del anterior programa son como sigue:

Un último ejemplo: Con base en los datos del anuario estadí stico de Antioquia sobre la composición profesoral en 10 universidades, veamos el uso de la función "legend" y del argumento "col" en el barplot:

```
datos.prof<-read.table("c:/graficos R/universidades.txt",
```

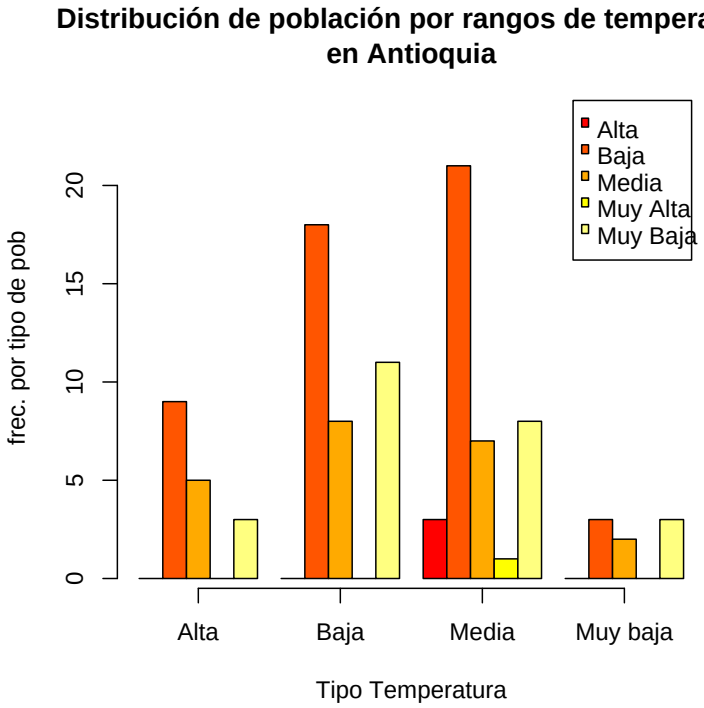

Figura 2.19: *Observe como la opción beside=T hace que las barras queden yuxtapuestas;las columnas (tipos de temperaturas) de la tabla de frecuencias aparecen representadas por cinco barras consecutivas (si los valores son todos no nulos), una por cada fila (tipo de población), en tanto que cada columna es separada de la siguiente por un pequeño espacio*

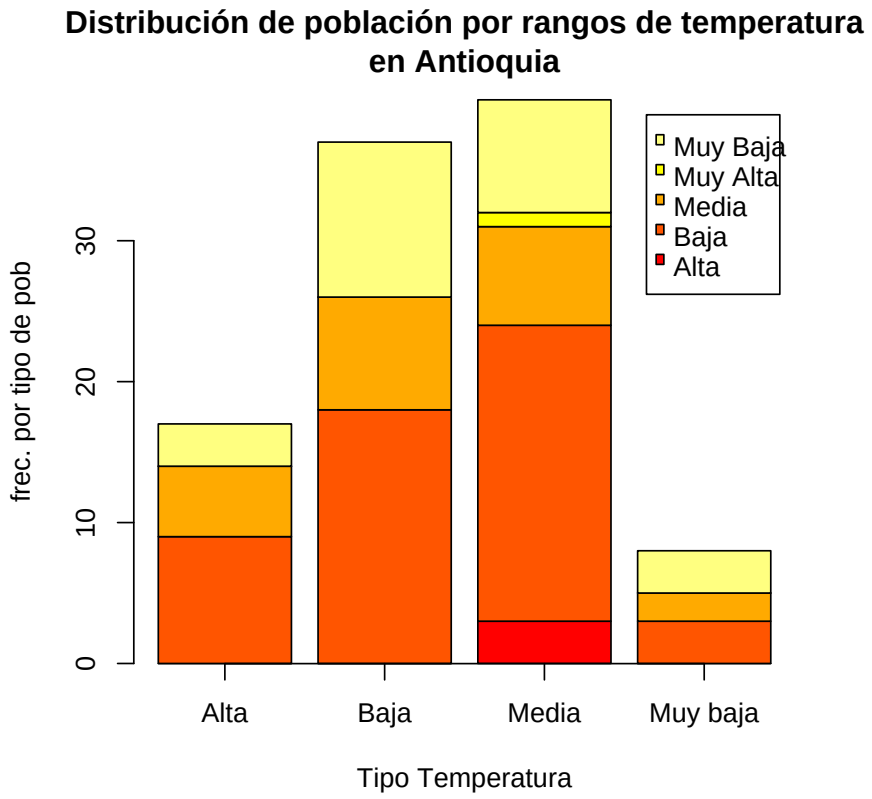

Figura 2.20: *Observe como la opción beside=FALSE hace que por cada columna (tipo de temperatura) de la tabla de frecuencias aparezca una barra en el gráfico dividida en franjas que representan las frecuencias de municipios en cada rango de población dentro de dicha categoría de temperatura. En comparación con el granterior, este último resulta poco apropiado para efectos de comparaciones*

```
header=T)
nombres.filas<-c('UdeA','UNal','EAFIT','EscIng','CES',
'UNAULA','UCC','UdeM',
'UPB','USanBu')
datos.prof2<-data.matrix(datos.prof[,2:7])
barplot(datos.prof2[,1:3],beside=T,space=c(0,0.5),xlim=c(0,35),
col=c('gray20','gray30','gray40','gray50','gray60','gray70',
'gray80','gray90','gray100'),
main='Composici\'on profesoral en 10 universidades de
Antioquia', xlab='Composici\'on profesoral',
ylab='Cant. por Universidad')
legend(30.8,1100,nombres.filas,fill=c('gray20','gray30',
'gray40','gray50',
'gray60','gray70','gray80','gray90','gray100'),cex=0.5)
barplot(datos.prof2[,1:3],beside=FALSE,legend.text=nombres.filas,
main='Composici\'on
profesoral en 10 universidades de Antioquia',
col=c('gray10','gray20','gray30','gray40','gray50','gray60',
'gray70','gray80','gray90','gray100'),xlim=c(0,5),
xlab='Composici\'on
profesoral',ylab='Cant. por Universidad')

#o bien

barplot(datos.prof2[,1:3],beside=FALSE,col=c(10:20),
main='Composici\'on profesoral en 10
universidades de Antioquia', xlim=c(0,5),
xlab='Composici\'on profesoral',ylab='Cant. por
Universidad') legend(4,5000,nombres.filas,fill=c(10:20))
```

Los gráficos resultantes son los siguientes:

2.3 Gráficos especiales en R

2.3.1 Gráfico de coordenadas polares "Polar plot"

Permite presentar datos en un plano de coordenadas polares, mediante la función que se especifica a continuación, :

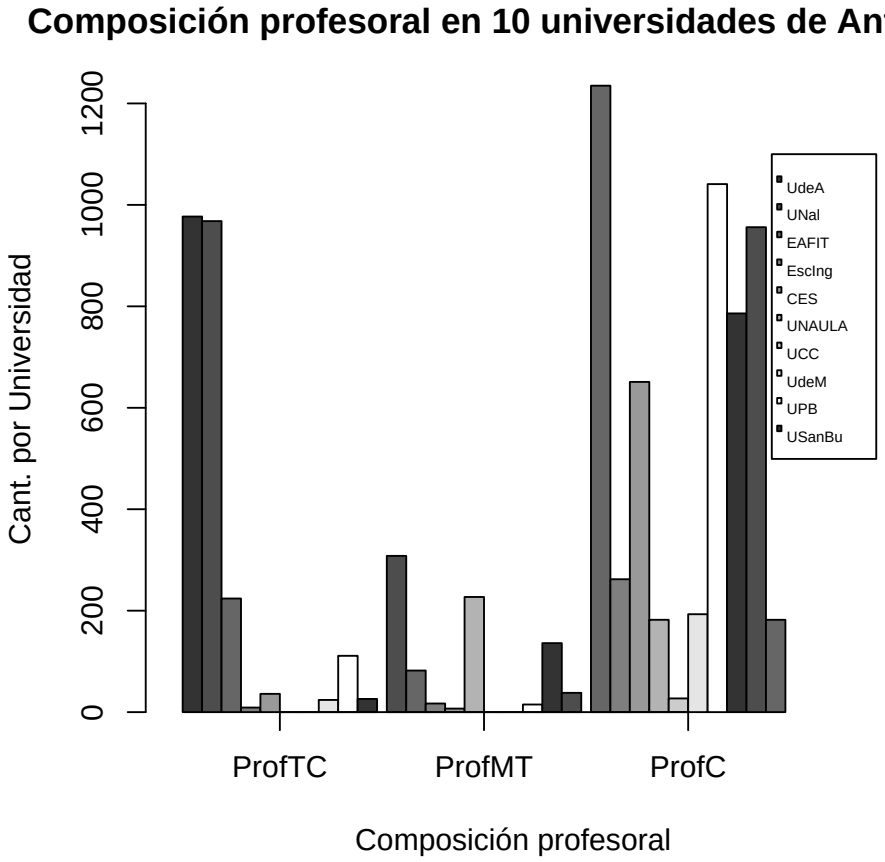

Figura 2.21: *Cuando el argumento legend.text no es especificado en le barplot, el gráfico producido resultaría sin leyenda. Sin embargo la función legend permite posteriormente agregar y especificar una leyenda para un gráfico, indicando las coordenadas x,y para la ubicación de la leyenda, el contenido de ésta y los colores para las cajas de la leyenda. Estos últimos deben ser iguales a los colores empleados en el argumento "col" del barplot*

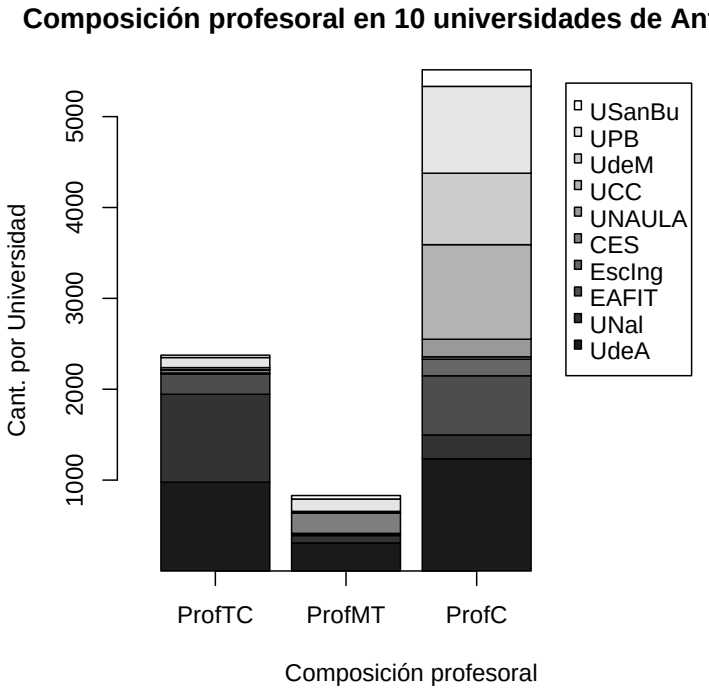

Figura 2.22: Observe el efecto en el color del argumento col=c('gray10','gray20','gray30','gray40','gray50','gray60', 'gray70','gray80','gray90','gray100')

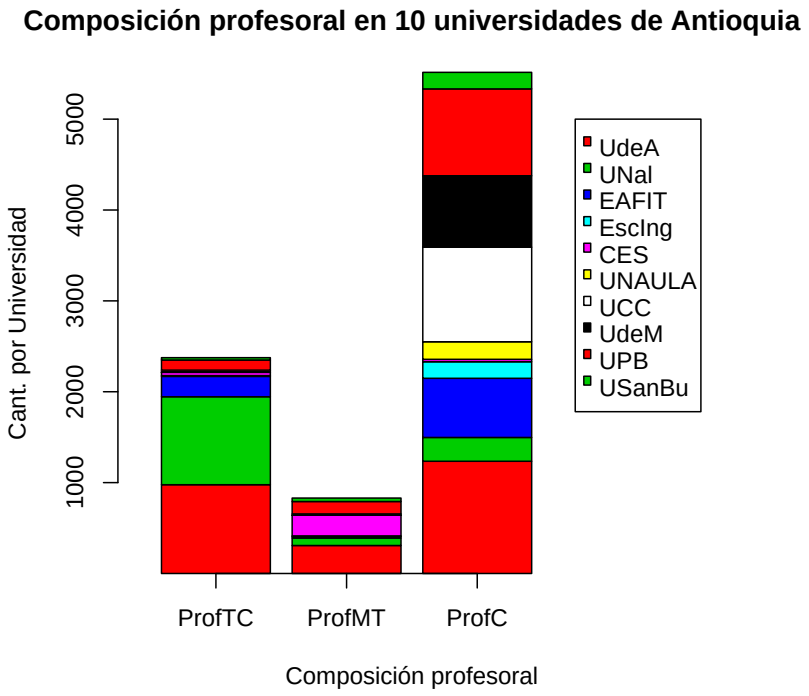

Figura 2.23: Observe el efecto en el color del argumento col=c(10:20)

Autor: Ross Ihaka, Department of Statistics, University of Auckland. Private Bag 92019, Auckland New Zealand:

- Asumiendo que r está dado en radianes

```
polar.plot <- function(r, theta,pch = NULL,col = NULL,
angle.axis = -90){
```

- Determinar el rango de valores.
- Elegir los puntos de corte en valores exactos los cuales se incluyen en el rango.

```
rpretty <- pretty(range(abs(r), 0, na.rm=TRUE))
rmax <- rpretty[length(rpretty)]
```

- Inicio de un nuevo gráfico.

```
plot.new()
```

- Fijar las coordenadas del gráfico.
- Elegir una región cuadrada la cual incluye un círculo suficientemente grande para incluir a todos los datos. El uso de asp = 1, permite que los círculos tengan tal apariencia aún cuando la ventana de gráficos sea redimensionada.

```
plot.window(xlim = c(-rmax, rmax),ylim = c(-rmax, rmax),
asp = 1)
```

- Dibujar una rejilla circular
    1. Los círculos son realmente polígonos con muchos vértices

2. El número “5” es elegido para que un cambio de dirección de menos de 5 grados aparezca suave.

```
grid <- seq(0, 2 * pi, length = 360 / 5 + 1)
for(rad in rpretty){
if(rad > 0)
lines(rad * cos(grid), rad * sin(grid), col = "gray")
}
```

- Dibujar una rejilla circular en gris. El uso de “12” da divisiones en secciones de 30 grados.

```
rad <- seq(0, 2 * pi, length = 12 + 1)[-1]
segments(0, 0, rmax * cos(rad), rmax * sin(rad), col = "gray")
```

- Etiquetar ejes básicos. Las etiquetas aparecen a lo largo del radio en ángulo “angle.axis” al eje x

```
text(rpretty[-1] * cos(angle.axis * pi / 180),
     rpretty[-1] * sin(angle.axis * pi / 180),
     rpretty[-1])
```

- Graficar los datos.

```
points(r * cos(theta),r * sin(theta),col = col, pch = pch)
}
```

- Ejemplo

```
group <- sample(3, 100, replace = TRUE)
theta <- 0.5 * rnorm(100) +  0.5 * group
r <- rnorm(100, 4)

polar.plot(r, theta, col = c("red","green4","blue")[group],
pch = 20)
```

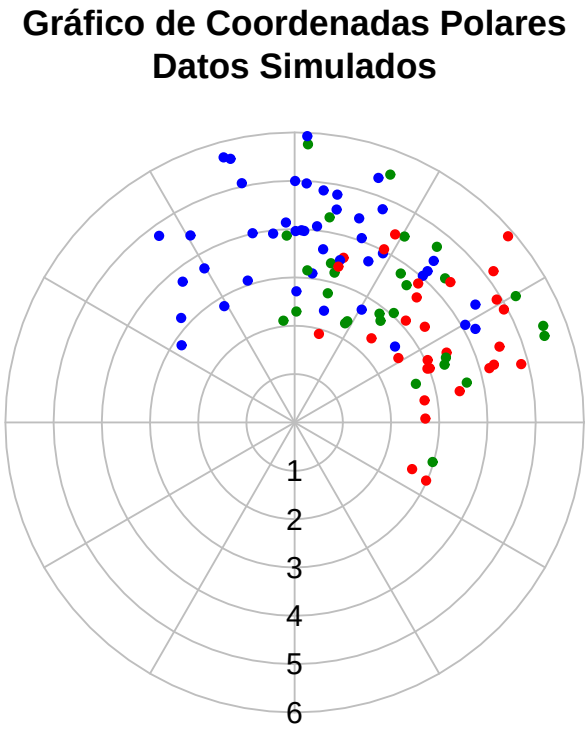

Figura 2.24: *Esta figura utiliza la función polar.plot aplicada sobre los datos simulados presentados en el ejemplo previo. Observe como este gráfico nos permite analizar la dispersión y a la vez identificar grupos o clusters en un conjunto de datos.*

2.3.2 Carta o Gráfico de Eventos

Este tipo de gráficos es común en estudios de tiempos de vida o de sobrevivencia. En el eje horizontal se describen los tiempos o períodos de observación y en el eje vertical las unidades o individuos cuyos tiempos de vida son observados. Obviamente no todas las unidades o individuos tendrán el mismo tiempo de vida y por otra parte, es posible que la observación sea realizada tan sólo hasta cierto tiempo t_0 (bien sea por diseño del experimento o por alguna otra circunstancia que impide seguir observando a un individuo u unidad), razón por la cual algunas de las unidades o individuos aún "sobreviven" al final del experimento, se dice que estos han sido censurados en t_0. Otra posible situación es que no todas las unidades (individuos) comienzan a ser observados en el mismo tiempo. La idea de este gráfico es indicar para cada caso los siguientes eventos: el tiempo de inicio de observación, el tiempo de fallo (muerte), o si fue censurado y además cualquier otro evento especial, como controles peíodicos. A continuación se presenta un posible programa en R para la elaboración de este tipo de gráficos y dos gráficos obtenidos a partir de éste:

- paciente corresponde a una variable numérica, pero tomada como un código para el paciente
- inicio es una variable numérica que indica el punto de inicio de las observaciones, si es cero significa que la fecha de comienzo de observación para un individuo coincide con la fecha de inicio del estudio
- días es el número de días totales de observación de un individuo
- censura es un vector de unos y ceros; el valor de 0 si la observacion fue censurada o de 1 si el individuo fue observado hasta el "fallo"
- especiales es una matriz; la columna 1 para indicar el número del individuo, la columna 2 para indicar el tiempo a partir del respectivo inicio en que ocurre el evento especial y la col 3 para indicar el tipo de evento especial. Para el caso se tienen tres eventos para E1 el valor es 1, para E2 es 2 y para E1,2 es 3. Si no ocurren eventos especiales, se omite en el argumento al llamar la función carta.eventos
- "a" es una variable que por defecto toma el valor de 0. Si existen eventos especiales debe especificarse como 1.

```
carta.eventos<-function(paciente,inicio,dias,censura,
especiales="NULL",a=0){
n<-length(paciente)
plot(inicio,paciente,xlim=c(-1,(max(dias+inicio)+1)),
ylim=c(-0.5,(n+1)),xlab="dias",pch=19)
for(i in 1:n){
temp1<-censura[i]
if(temp1==0)temp1<-4
else temp1<-0
points((inicio[i]+dias[i]),paciente[i],pch=temp1)
segments(inicio[i],paciente[i],(inicio[i]+dias[i]),
paciente[i])
}
if(a==1){
m<-nrow(especiales)
for(j in 1:m){
temp2<-especiales[j,3]
if(temp2==1)temp2<-17
else{
if(temp2==2)temp2<-15
else temp2<-13
}
temp3<-0
for(i in 1:n){
temp3[especiales[j,1]==i]<-inicio[i]
}
points((temp3+especiales[j,2]),especiales[j,1],pch=temp2)
}
}
x1<--0.5
y<-0.5
x2<-max(dias+inicio)/5-1
x3<-2*max(dias+inicio)/5-2
x4<-3*(max(dias+inicio))/5-3
x5<-4*(max(dias+inicio))/5-4
x6<-max(dias+inicio)-4
legend(x1,y,c("inicio"),bty="n",cex=0.7,pch=19)
legend(x2,y,c("fallo"),bty="n",cex=0.7,pch=4)
```

```
legend(x3,y,c("E1"),bty="n",cex=0.7,pch=17)
legend(x4,y,c("E2"),bty="n",cex=0.7,pch=15)
legend(x5,y,c("E1,2"),bty="n",cex=0.7,pch=13)
legend(x6,y,c("censura"),bty="n",cex=0.7,pch=0)
}

paciente<-c(1,2,3,4)
inicio<-c(0,0,1,5)
dias<-c(20,20,15,12)
censura<-c(1,1,0,0)
especiales<-matrix(c(1,5,1,1,15,2,2,5,1,2,15,2,4,10,3),
ncol=3,byrow=T)
> especiales
     [,1] [,2] [,3]
[1,]    1    5    1
[2,]    1   15    2
[3,]    2    5    1
[4,]    2   15    2
[5,]    4   10    3

fig1<-carta.eventos(paciente,inicio,dias,censura,
especiales,1)
fig2<-carta.eventos(paciente,inicio,dias,censura)
```

2.3.3 Pirámide Poblacional

Este tipo de gráficos permite presentar en forma comparativa, la distribución de la población, por rangos de edad, según dos grupos, por ejemplo, hombres y mujeres. Una pirámide poblacional puede construirse en R a partir de las dos siguientes funciones:

```
piramide1<-function(hombres,mujeres,amplitud,escalax,
edadmax,region){
max1<-max(c(hombres,mujeres))
n<-length(hombres)
```

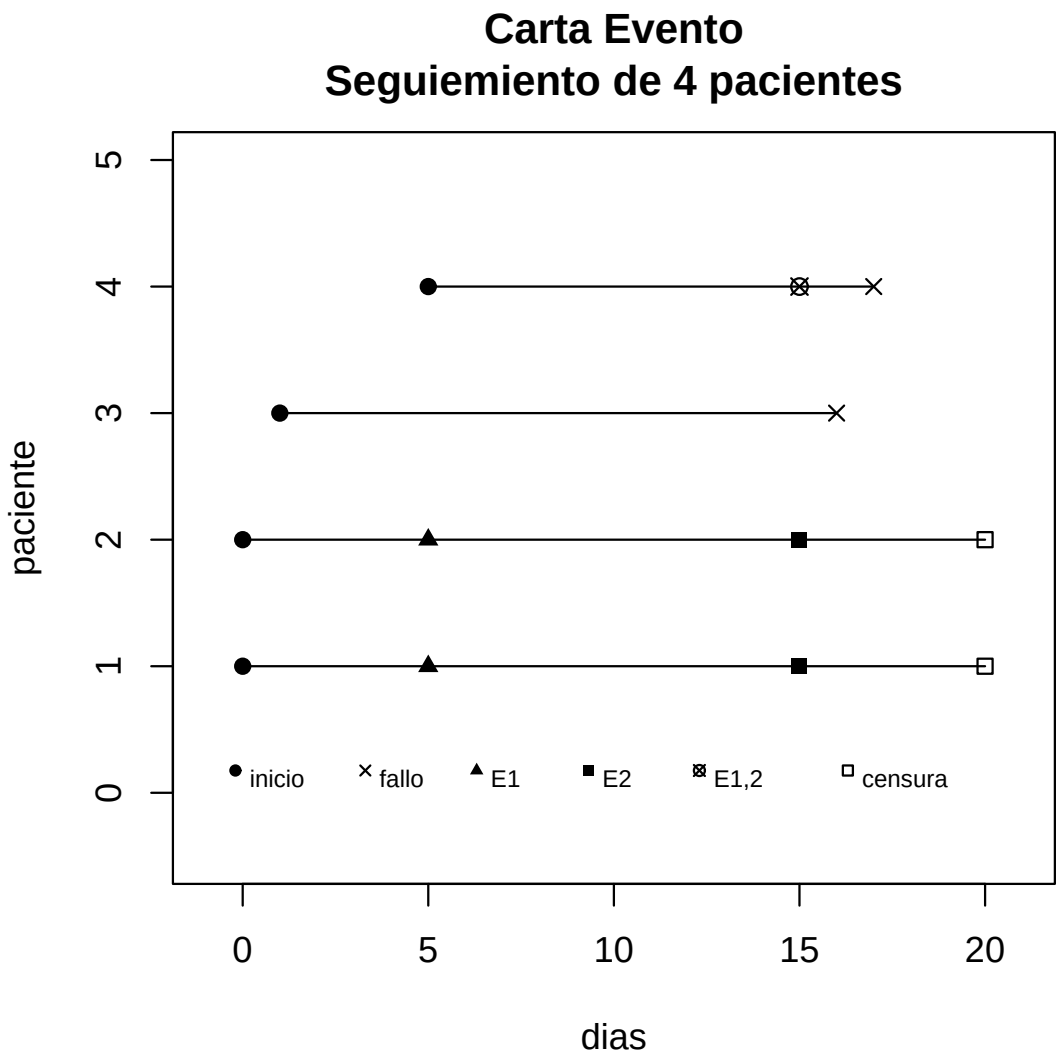

Figura 2.25: *Esta gráfica presenta el gráfico de eventos considerando la ocurrencia de los eventos especiales E1, E2, y E1,2 para los datos supuestos en un experimento en el cual se observan los tiempos de vida de cuatro individuos sometidos a un tratamiento específico. Es obtenida con* $fig1 <- carta.eventos(paciente, inicio, dias, censura, especiales, 1)$

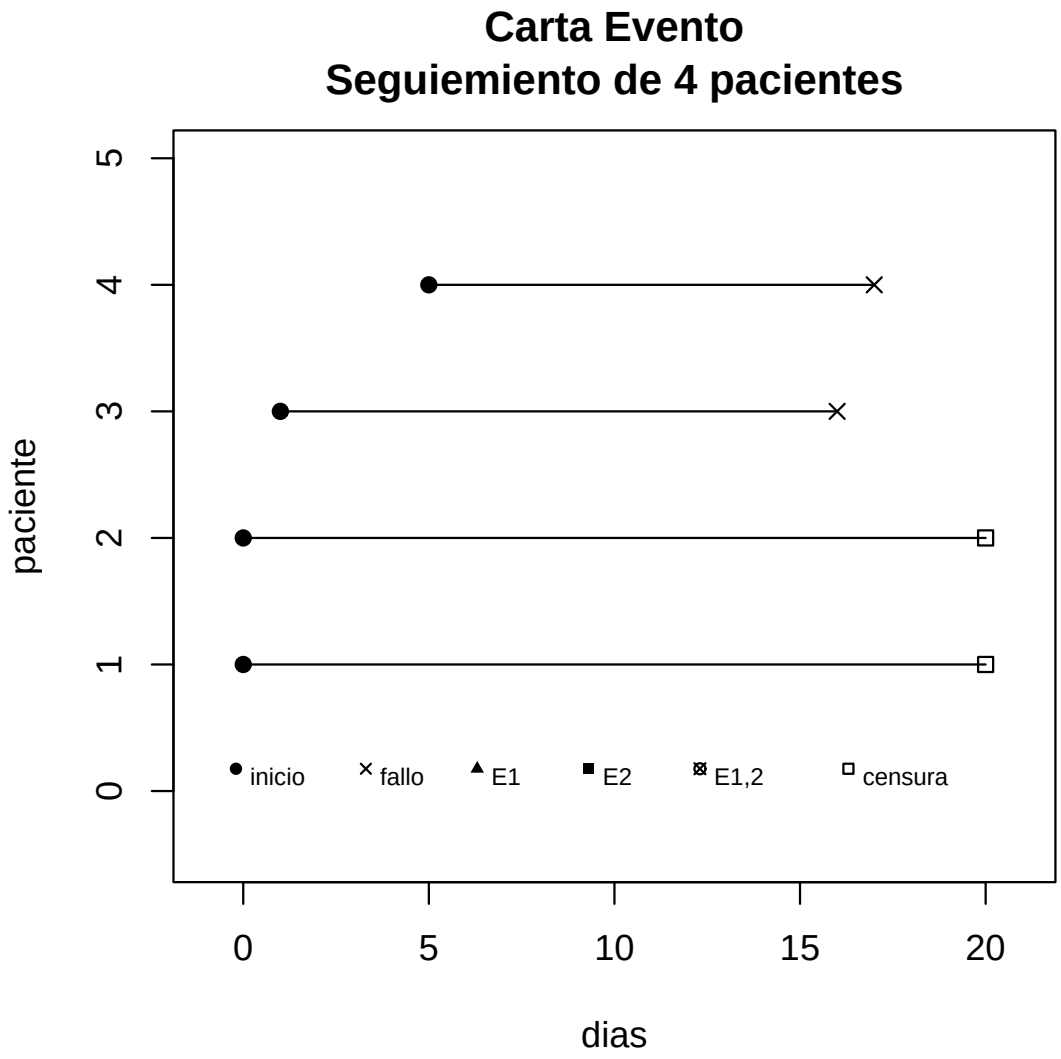

Figura 2.26: *Esta gráfica presenta el gráfico de eventos sin considerar la ocurrencia de los eventos especiales E1, E2, y E1,2 para los mismos datos del ejemplo previo. Es obtenida con* $fig2 <- carta.eventos(paciente, inicio, dias, censura)$

```
#Simetrizar la figura, ubicando datos de las mujeres
#a la izquierda de los hombres,
#Utilizar el factor de escala 'excalax' para el eje x:

min.x<--(max1%/%escalax+1)*escalax
max.x<-(max1%/%escalax+1)*escalax


#Abrir una ventana para el gráfico

plot(0,0,type="n",xaxt='n',yaxt='n',ylim=c(0,edadmax+5),
xlim=c(min.x,max.x),xlab="",ylab="")

#el anterior plot se utiliza para abrir
#la ventana de gráficos
#pero observe type="n",
#con ello se indica que no despliegue
#ningún gráfico y
#xaxt='n',yaxt='n' hace que los
#ejes no sean dibujados

#dibujar ejes y etiquetarlos:

ejex1<-seq(0,max1,by=escalax)
ejex2<--ejex1[order(-ejex1)]
ejex<-c(ejex2,ejex1)
axis(1,at=ejex,labels=as.character(abs(ejex)),
cex.axis=0.8,las=2)
ejey<-c(seq(0,edadmax,by=amplitud))
axis(2,at=ejey,labels=as.character(ejey),cex.axis=0.8,las=2)

for(i in 1:n){
x1<-0
x2<-hombres[i] x3<--mujeres[i]

#permitirá que los datos de las mujeres queden a la
#izquierda del grafico.
```

```
#las dos siguientes lineas
#definen los ticks de la escala de edad según

#la amplitud (igual) de los intervalos elegidos:

y1<-(i-1)*amplitud
y2<-y1+amplitud

#para dibujar la distribución de los hombres
#llamamos la función rect del R:

rect(x1,y1,x2,y2,col='red')

#para dibujar la distribución de las mujeres en el
#mismo grafico, llamamos la función
#rect de nuevo:

rect(x1,y1,x3,y2,col='blue')

#en la función rect las coordenadas del rectángulo
#se dan en orden x izquierdo, e
#inferior, x derecho, y superior

}

#Ubicar leyendas y titulo:

x.l1<--max1/16-1.5*escalax
x.l2<-max1/16+escalax
title(main=paste("Pirámide
Poblacional",sep="\n",region),ylab="edad")

legend(x.l1,edadmax+5,"mujeres",bty="n",xjust=1)
legend(x.l2,edadmax+5,"hombres",bty="n")
}
```

Ejemplo 1: Pirámide poblacional para la ciudad de Medellín según censo de 1993:

```
pob<-read.table("c:/graficosr/datosgraficos/pobmedellin.txt",
header=T)
> pob
   grupoedad hombres mujeres
1        0-4   77674   75632

2        5-9   79523   77809

3      10-14   78607   79028

4      15-19   64907   76456

5      20-24   68331   86469

6      25-29   76046   92316

7      30-34   71487   84937

8      35-39   55728   69383

9      40-44   44853   54049

10     45-49   33646   40652

11     50-54   27500   34709

12     55-59   20762   27312

13     60-64   18981   26097

14     65-69   13516   18318

15     70-74    9510   13770
```

```
16     75-79    6190    9268

17     80-84    3490    5749

18      85-+    2430    4835

mujeres<-pob[,3]
hombres<-pob[,2]
max(max(mujeres),max(hombres))
[1] 92316

#Por lo anterior un factor de escala
#apropiada en eje x es de 10000
amplitud<-5
escalax<-10000
edadmax<-90
region<-"Medellín"
fig<-piramide1(hombres,mujeres,amplitud,escalax,
edadmax,region)
```

Ejemplo 2: Otro posible presentación de una pirámide poblacional puede ser también como la que ofrece la siguiente función:

```
piramide2<-function(hombres,mujeres,amplitud,escalax,
edadmax,region){
max1<-max(c(hombres,mujeres))
min.x<--(max1%/%escalax+1)*escalax
max.x<-(max1%/%escalax+1)*escalax
n<-length(hombres)
plot(0,0,type="n",xaxt='n',yaxt='n',ylim=c(0,edadmax+5),
xlim=c(min.x-2*escalax,max.x+2*escalax),xlab="",ylab="")

ejex1<-seq(2*escalax,max.x+2*escalax,,by=escalax)
ejex2<--ejex1[order(-ejex1)]
ejex<-c(ejex2,ejex1)
ejexe1<-seq(0,max.x,by=escalax)
```

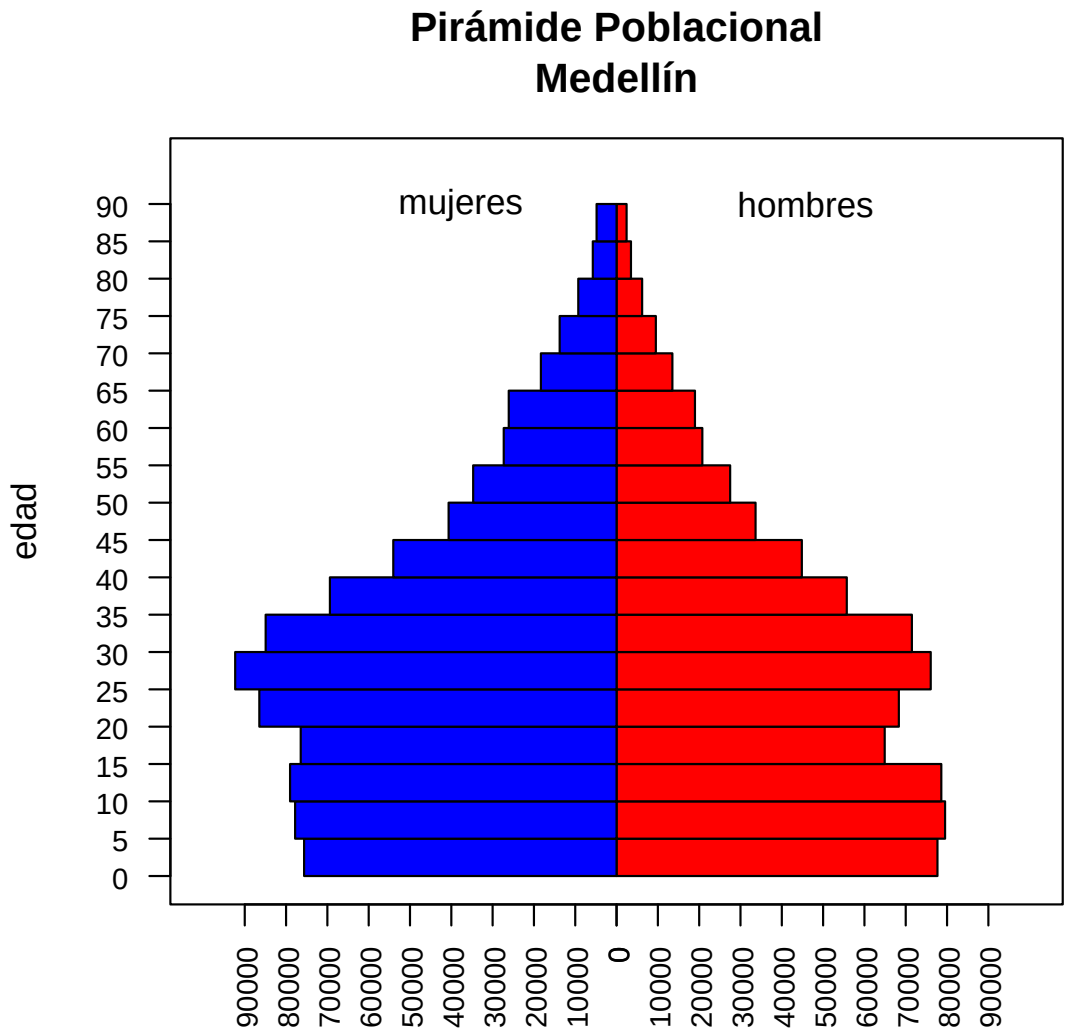

Figura 2.27: *Pirámide poblacional con base en datos del último censo efectuado en octubre de 1993, construída con la función 'piramide1'.*

```
ejexe2<--ejexe1[order(-ejexe1)]
ejexe<-c(ejexe2,ejexe1)
axis(1,at=ejex,labels=as.character(abs(ejexe)),
cex.axis=0.8,las=2)

ejey<-c(seq(0,edadmax,by=amplitud))
axis(2,at=ejey,labels=as.character(ejey),
cex.axis=0.6,las=2,pos=0)

for(i in 1:n){
x1<-2*escalax
x2<-hombres[i]+2*escalax
x3<--mujeres[i]-2*escalax
y1<-(i-1)*amplitud
y2<-y1+amplitud

rect(x1,y1,x2,y2,col='red')
rect(-x1,y1,x3,y2,col='blue')
}
x.l1<--max1/16-2.5*escalax
x.l2<-max1/16+2*escalax

title(main=paste("Pirámide Poblacional",sep="\n",region))
legend(x.l1,edadmax+5,"mujeres",bty="n",xjust=1)
legend(x.l2,edadmax+5,"hombres",bty="n")
legend(0,edadmax+10,"edad",bty='n',xjust=0.5,adj=c(0.5,0.5)) }
```

Que aplicada sobre los datos de población para la ciudad de Medellín:

```
amplitud<-5
escalax<-10000
edadmax<-90
region<-"Medellín"
fig<-piramide2(hombres,mujeres,amplitud,escalax,
edadmax,region)
```

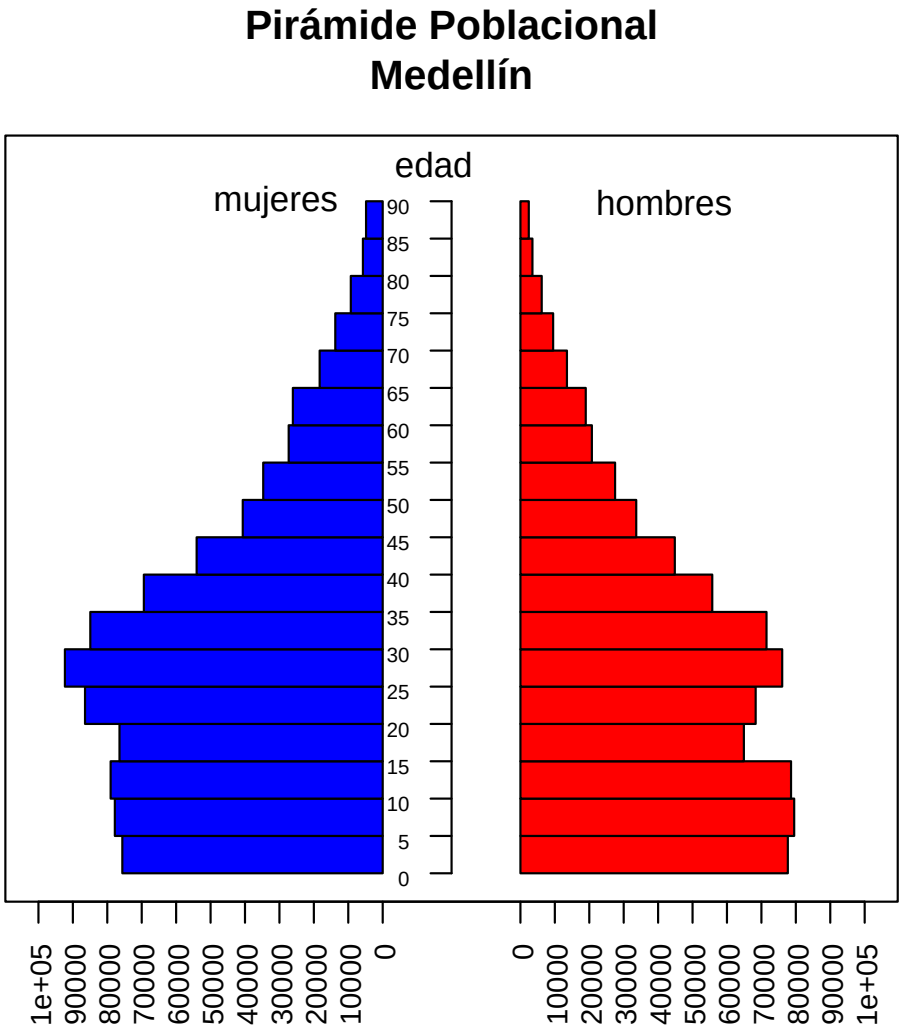

Figura 2.28: *Pirámide poblacional obtenida con la función 'piramide2'.*

Capítulo 3

Gráficos Multivariables en R

3.1 Gráficos de Dispersión

Es tal vez el más antiguo de los gráficos multivariables. Está limitado a la presentación de dos variables, aunque se pueden realizar modificaciones de tal forma que nos permita incluir más.

En R obtenemos este gráfico mediante la función plot:

```
plot(x, ...)
plot(x, y, xlim=range(x), ylim=range(y), type="p",
     main, xlab, ylab, ...)
plot(y ~ x, ...)
```

Donde:

- x: Objeto que contiene las coordenadas de los puntos del gráfico.
- y: Las coordenadas y de los puntos, sin embargo resulta innecesario cuando el objeto x tiene la estructura apropiada para dar ambas coordenadas
- xlim, ylim: Los rangos para los ejes x e y respectivamente.
- type: Para especificar el tipo de gráfico, así :
    1. "p" para puntos, lo cual es por defecto
    2. "l" para trazar líneas entre los puntos
    3. "b" para puntos y líneas

4. “o” para puntos y líneas superpuestos
5. “h” para un histograma pero en vez de rectángulos traza líneas verticales
6. “s” Para gráficos en escala
7. “S” Otros gráficos en escala
8. “n” Para que no aparezca el gráfico.

- main: Para titular al gráfico.
- xlab, ylab: Para etiquetar a los ejes.
- ...: Otros parámetros gráficos.

En el argumento “type” se especifica igual a “p” para representar las observaciones por puntos, aunque este es el valor por defecto y por tanto no es necesario especificarlo. En el siguiente ejemplo se observa el uso de esta función:

```
preciosr9<-matrix(scan("c:/graficosr/graficos/r9.txt"),
ncol=13,byrow=T)
ano<-preciosr9[1,]
valor<-preciosr9[2,]
plot(ano,valor,main='Precio de Oferta Renault 9
vs. Año',xlab='año',ylab='valor',pch=19)
```

3.2 Matrices de Dispersión

Las matrices de dispersión proporcionan un método simple de presentar las relaciones entre pares de variables. Consiste en una matriz donde cada entrada presenta un gráfico de dispersión sencillo. Un inconveniente es que si tenemos muchas variables el tamaño de cada entrada se reduce demasiado impidiendo ver con claridad las relaciones entre los pares de variables.
La celda (i,j) de una matriz de dispersión contiene el scatterplot de la columna i versus la columna j de la matriz de datos. Este gráfico puede obtenerse mediante:

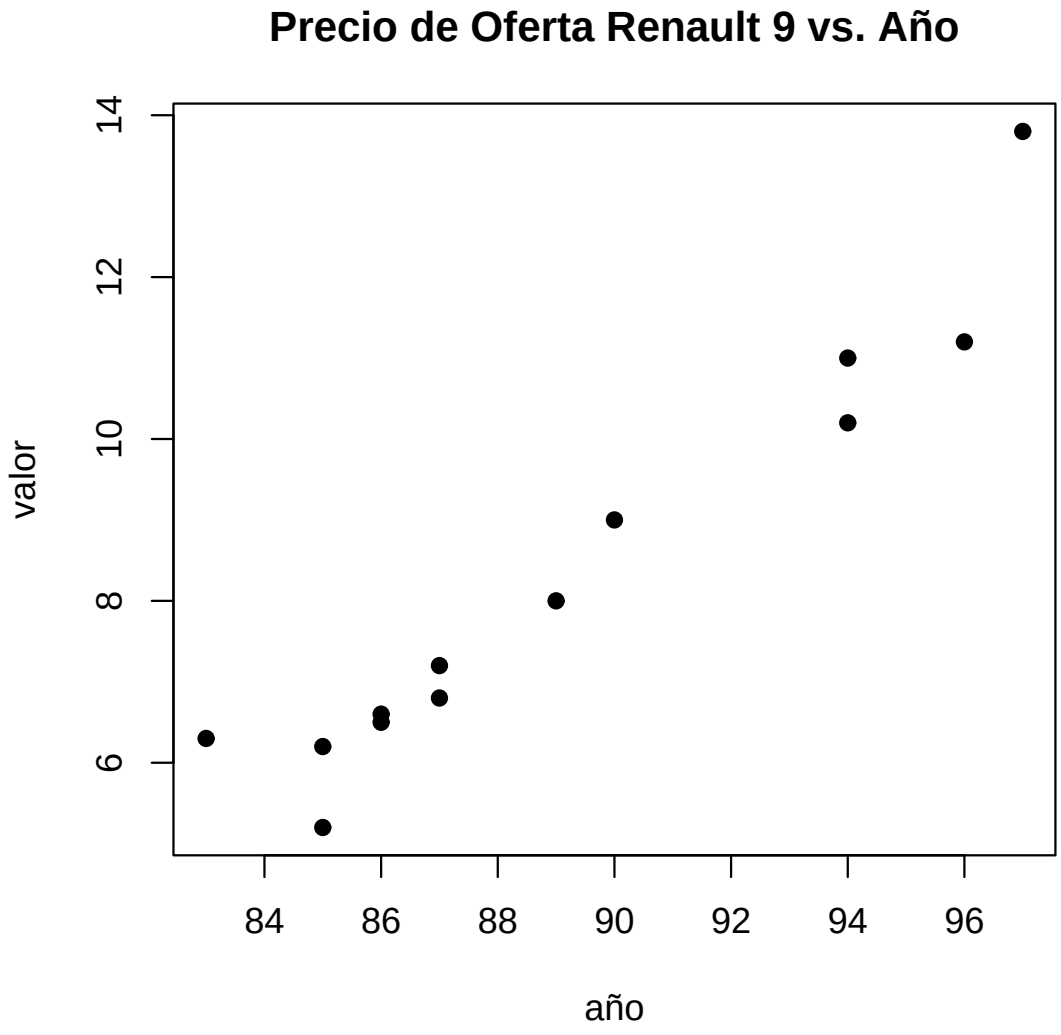

Figura 3.1: *Esta gráfica presenta información sobre el precio de oferta de carros Renault 9 vs. el año del carro aparecidos en los avisos clasificados de un periódico local.*

```
pairs(x, ...)
pairs.default(x, labels = colnames(x), panel = points, ...,
lower.panel = panel, upper.panel = panel,
diag.panel = NULL, text.panel = textPanel,
label.pos = 0.5 + has.diag/3,
cex.labels = NULL, font.labels = 1,
row1attop = TRUE)
```

Por ejemplo:

```
pies<-matrix(scan("c:/graficosr/graficos/dedos.dat"),
ncol=6,byrow=T)

#x1:longitud max de
la huella
#x2:amplitud max de la huella
#x3:amplitud max del talon #x4:longitud max del
dedo mayor
#x5:amplitud max del dedo mayor
#x6:sexo 1: hombre, 0: mujer

pairs(pies,labels=c("x1","x2","x3","x4","x5","x6"),
main='Matriz de dispersi\'on\n \npara
medidas de pies,cex.main=0.8)
```

Los argumentos de esta función corresponden a:

- x: Las coordenadas de puntos dados como columnas de una matriz.
- labels: Un vector para identificar los nombres de las columnas. Por defecto toma los nombres de columnas de la matriz x.
- panel: Para especificar una función: function(x,y,...), a ser usada para determinar los contenidos de los paneles o gráaficos componentes. Por defecto es points, indicando que se graficarán los puntos de los pares de datos. También es posible utilizar aquí otras funciones prediseñadas.
- ...: Indica que es posible agregar otros parámetros gráficos, tales como pch y col, con los cuales puede especificarse un vector de símbolos y colores a ser usados en los scatterplots.

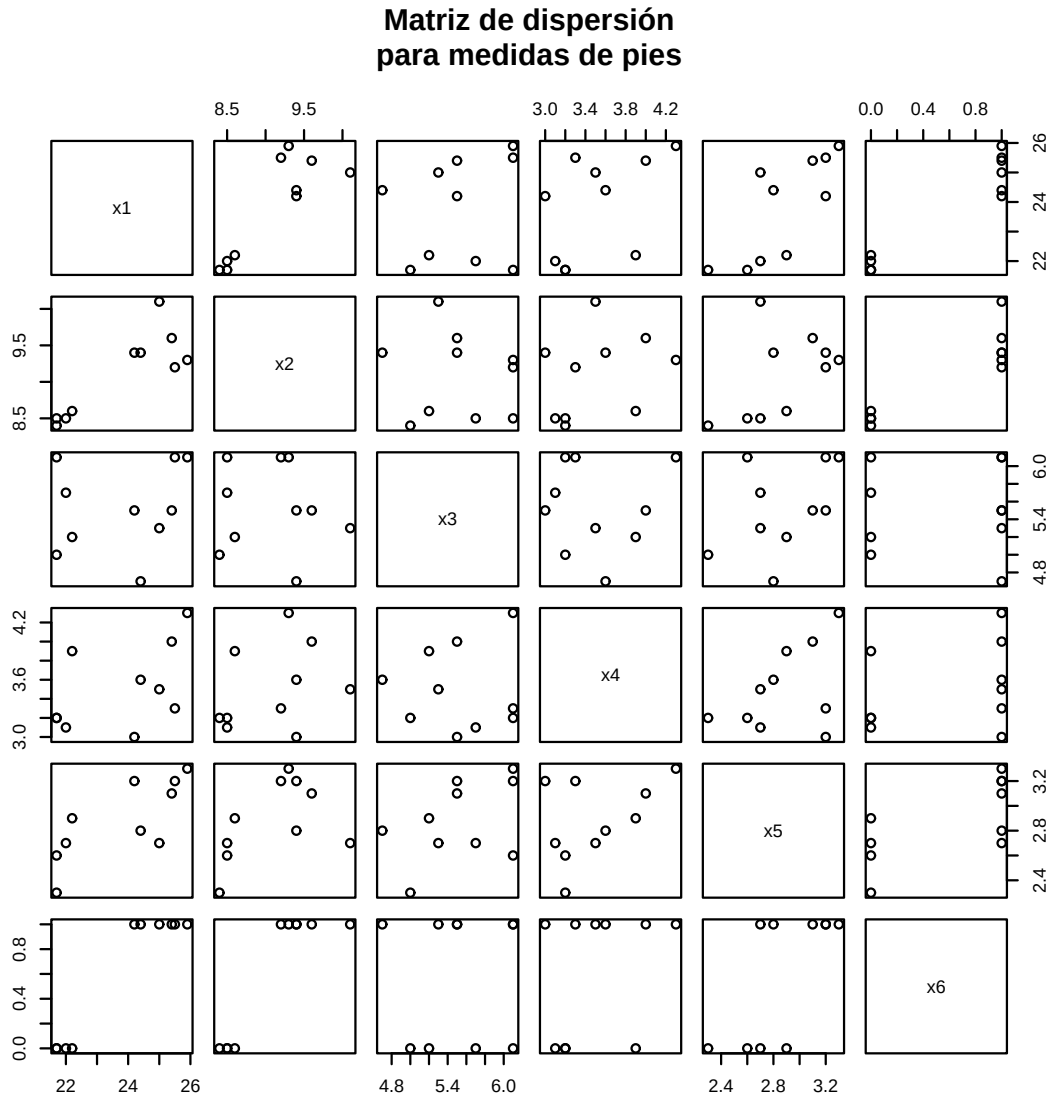

Figura 3.2: *Esta gráfica corresponde a los cruces de las variables medidas en los pies de varias personas.*

- lower.panel, upper.panel: Para especificar funciones por separado a ser empleados en los paneles abajo y arriba de la diagonal respectivamente.
- diag.panel: Es opcional. Permite que una función: function(x, ...) sea empleada para los paneles de la diagonal.
- text.panel: Es opcional. Permite que una función: function(x, y, labels, cex, font, ...) sea aplicada a los paneles de la diagonal.
- label.pos: Para especificar la posición y de las etiquetas en el text panel.
- cex.labels, font.labels: Parámetros para la expansión de caracteres y fuentes a usar en las etiquetas de las variables.
- row1attop: Parámetro lógico con el cual se especifica si el gráfico para especificar si el diseño lucirá como una matriz con su primera fila en la parte superior o como un gráfico con fila uno en la base. Por defecto es lo primero.

Autores:

- Enhancements for R 1.0.0 contributed by Dr. Jens
- Oehlschlaegel-Akiyoshi and R-core members.

Las matrices de dispersión pueden variarse para presentar información diversa de modo gráfico sobre un mismo conjunto de datos multivariados. El ejemplo siguiente tomado de a ayuda del R ilustra algunas de estas posibilidades:

```
data(USJudgeRatings)
panel.hist <- function(x, ...)
{
usr <- par("usr"); on.exit(par(usr))

#para definir regi\'on de graficiaci\'on

par(usr = c(usr[1:2], 0, 1.5) )

#para obtener una lista que guarde las
```

```
#marcas de clase y conteos en cada una:

h <- hist(x, plot = FALSE)
breaks <- h$breaks;
nB <- length(breaks)
y <- h$counts; y <- y/max(y)

rect(breaks[-nB], 0, breaks[-1], y, col="cyan", ...)

#para dibujar los histogramas

}
pairs(USJudgeRatings[1:5], panel=panel.smooth, cex = 1.5,
pch = 19, bg="light blue",
diag.panel=panel.hist, cex.labels = 1, font.labels=1)

par(oma=c(1,1,1,1),new=T,font=2,cex=0.5)
mtext(outer=T,"Matriz de dispersión con
Histograma",side=3)
```

veamos ahora cómo dibujar boxplots en la diagonal de la matrix de dispersión:

```
data(USJudgeRatings)
panel.box <- function(x, ...)
{
usr <- par("usr",bty='n'); on.exit(par(usr))
par(usr = c(-1,1, min(x)-0.5, max(x)+0.5))
b<-boxplot(x,plot=FALSE)
whisker.i<-b$stats[1,]
whisker.s<-b$stats[5,]
hinge.i<-b$stats[2,]
mediana<-b$stats[3,]
hinge.s<-b$stats[4,]
rect(-0.5, hinge.i, 0.5,mediana,...,col='grey')
segments(0,hinge.i,0,whisker.i,lty=2)
```

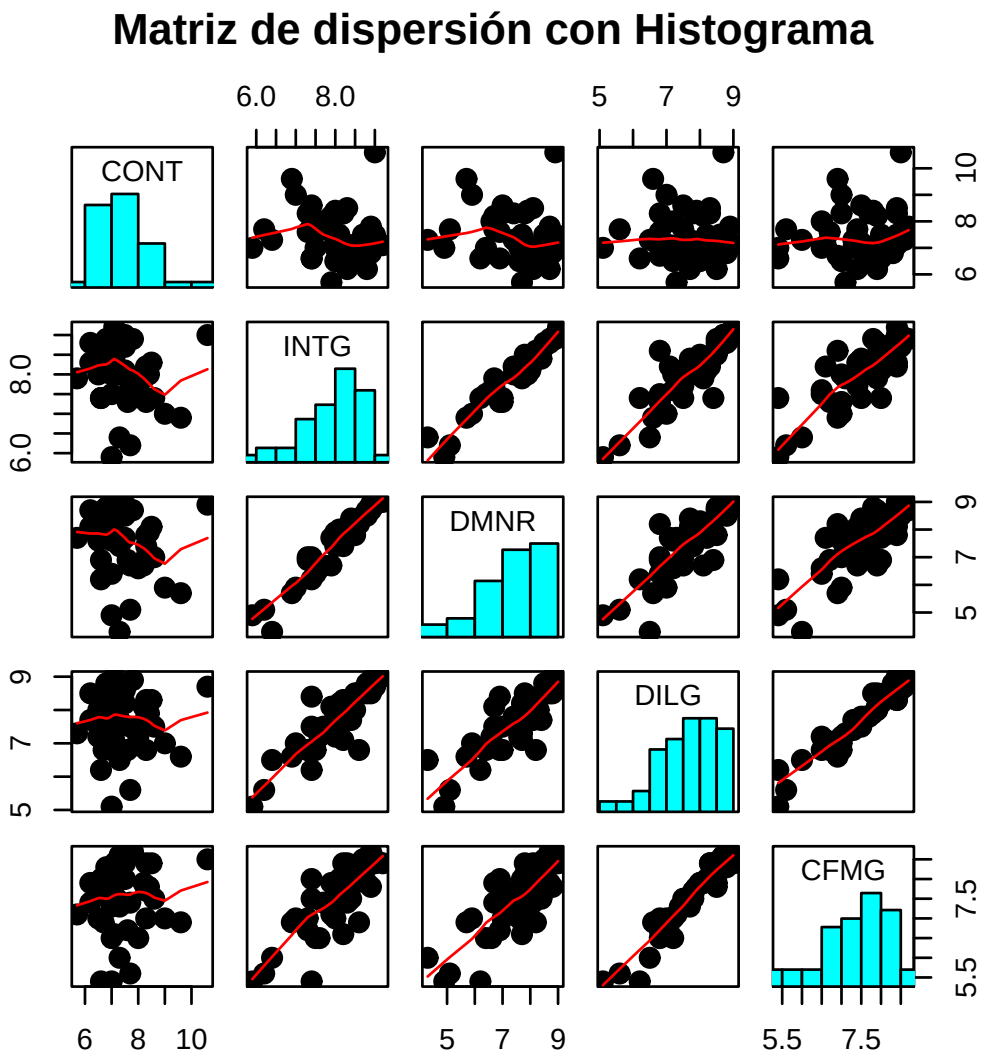

Figura 3.3: *Observe como al invocar la función pairs, el argumento panel se especifica igual a panel.smooth para obtener una curva de suavizamiento ajustada a los datos, y en el argumento diag.panel se indica utilizar la función panel.hist.*

```
segments(-0.1,whisker.i,0.1,whisker.i)
rect(-0.5, mediana, 0.5,hinge.s,...,col='grey')
segments(0,hinge.s,0,whisker.s,lty=2)
segments(-0.1,whisker.s,0.1,whisker.s)
}
pairs(USJudgeRatings[1:5],panel=panel.smooth,
cex = 1, pch = 19, bg="light blue",
diag.panel=panel.box, cex.labels = 0.8, font.labels=0.8)

par(oma=c(1,1,1,1),new=T,font=2,cex=0.5)
mtext(outer=T,"Matriz de dispersión con
Boxplot",side=3)
```

Obsérvese que cuando la función box es invocada pero el argumento plot se especifica como FALSE, los valores básicos del gráfico pueden ser guardados en un objeto matricial. Por ejemplo:

```
b<-boxplot(x,plot=FALSE)
whisker.i<-b$stats[1,]
hinge.i<-b$stats[2,]
mediana<-b$stats[3,]
hinge.s<-b$stats[4,]
whisker.s<-b$stats[5,]
```

stats es una matrix cuyas columnas contienen, el extremo del bigote inferior, la base inferior del box, la mediana, la base superior y el extremo del bigote superior para cada una de las columnas del objeto "x".

En el ejemplo que se presenta a continuación, se crea una función que calcula los valores de los coeficientes de correlación entre un par de variables, la cual luego es aplicada en un pairs para el upper.panel:

```
data(iris)
panel.cor <- function(x, y, digits=2, prefix="", cex.cor)
{
usr <- par("usr"); on.exit(par(usr))
```

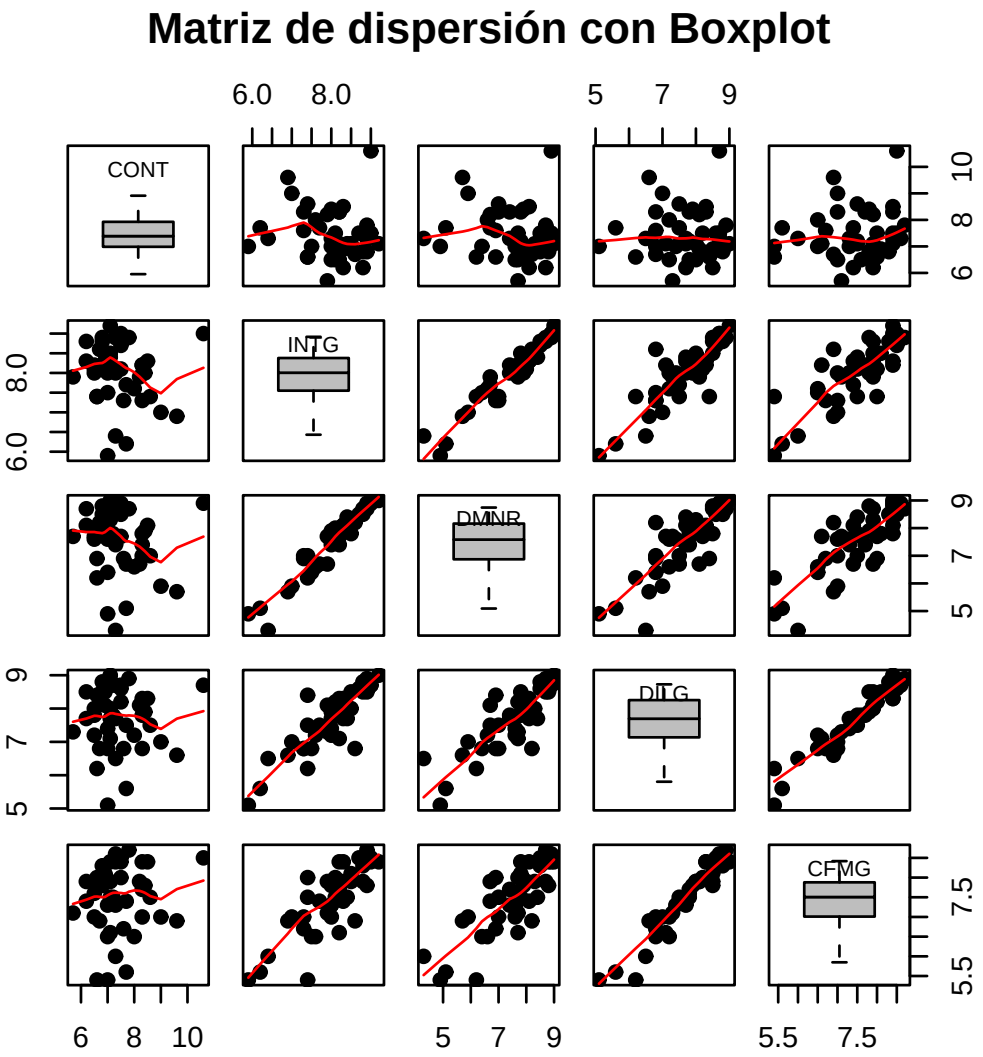

Figura 3.4: *En esta ocasión el argumento diag.panel del pairs se especifica igual a la función panel.box.*

```
par(usr = c(0, 1, 0, 1))
r <- abs(cor(x, y))
txt <- format(c(r, 0.123456789), digits=digits)[1]
txt <- paste(prefix, txt, sep="")
if(missing(cex.cor))
cex <- 0.8/strwidth(txt)
text(0.5, 0.5, txt, cex = cex)
}
pairs(iris[1:4], lower.panel=panel.smooth,
upper.panel=panel.cor)

par(oma=c(1,1,1,1),new=T,font=2,cex=0.5)
mtext(outer=T,"Matriz de dispersión con
correlaciones",side=3)
```

Otras modificaciones sencillas pero que permiten una mejor perspectiva acerca de las relaciones entre variables, es el uso de un vector de colores o de símbolos en los scatterplots de la matriz para el caso que exista un factor cuyos niveles puede influir en dichas relaciones. Veamos: En la base de datos "iris", las observaciones están clasificadas según un factor denominado "especie":

```
data(iris)
pairs(iris[1:4], main = "Data(iris) -- 3 Especies",
pch = 21,bg = c("red",
"green3", "blue")[codes(iris$Species)])

temp<-rep(NA,3)
temp[iris$Species=="setosa"]<-19
temp[iris$Species=="versicolor"]<-24
temp[iris$Species=="virginica"]<-12
pairs(iris[1:4],pch = temp)
par(oma=c(1,1,1,1),new=T,font=2,cex=0.5)

mtext(outer=T,"Matriz de dispersión con distinción de
muestras",side=3)
```

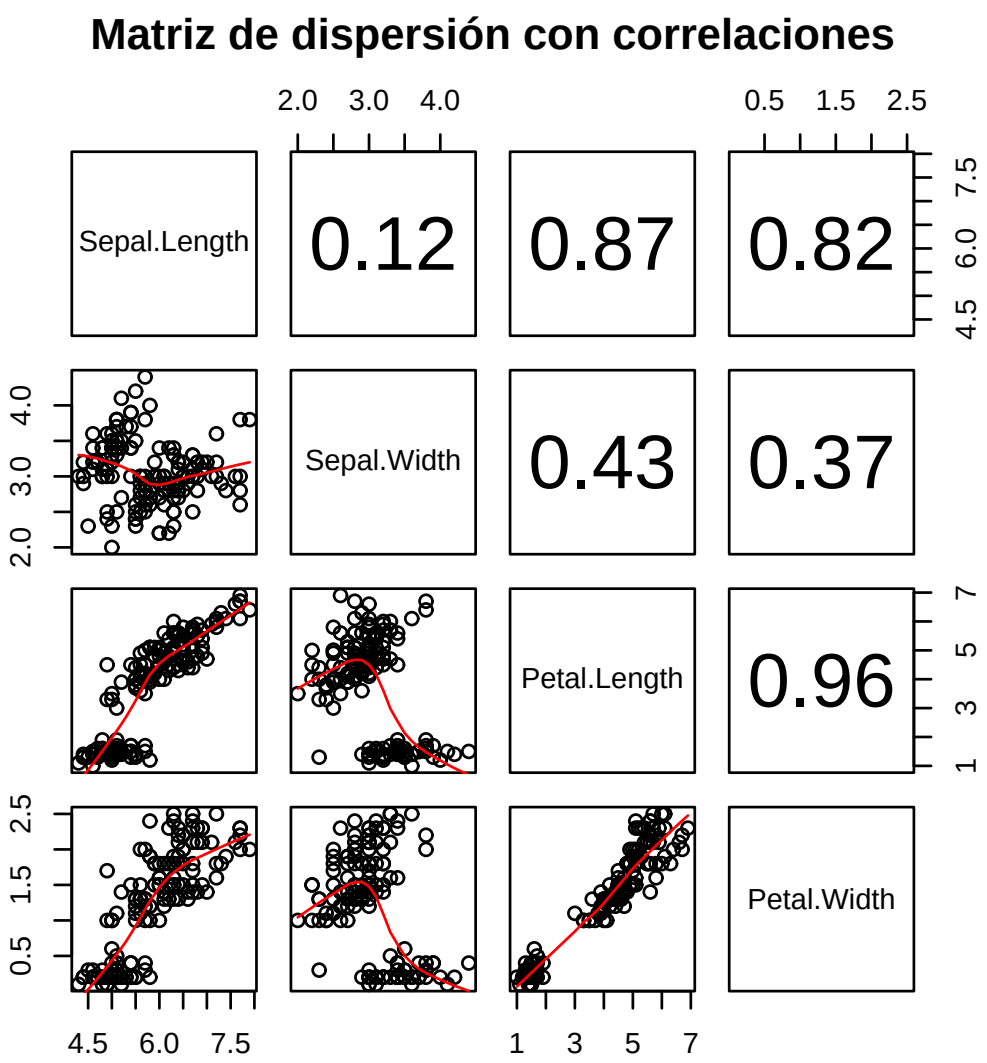

Figura 3.5: *En esta ocasión el argumento upper.panel del pairs se especifica igual a la función panel. ..*

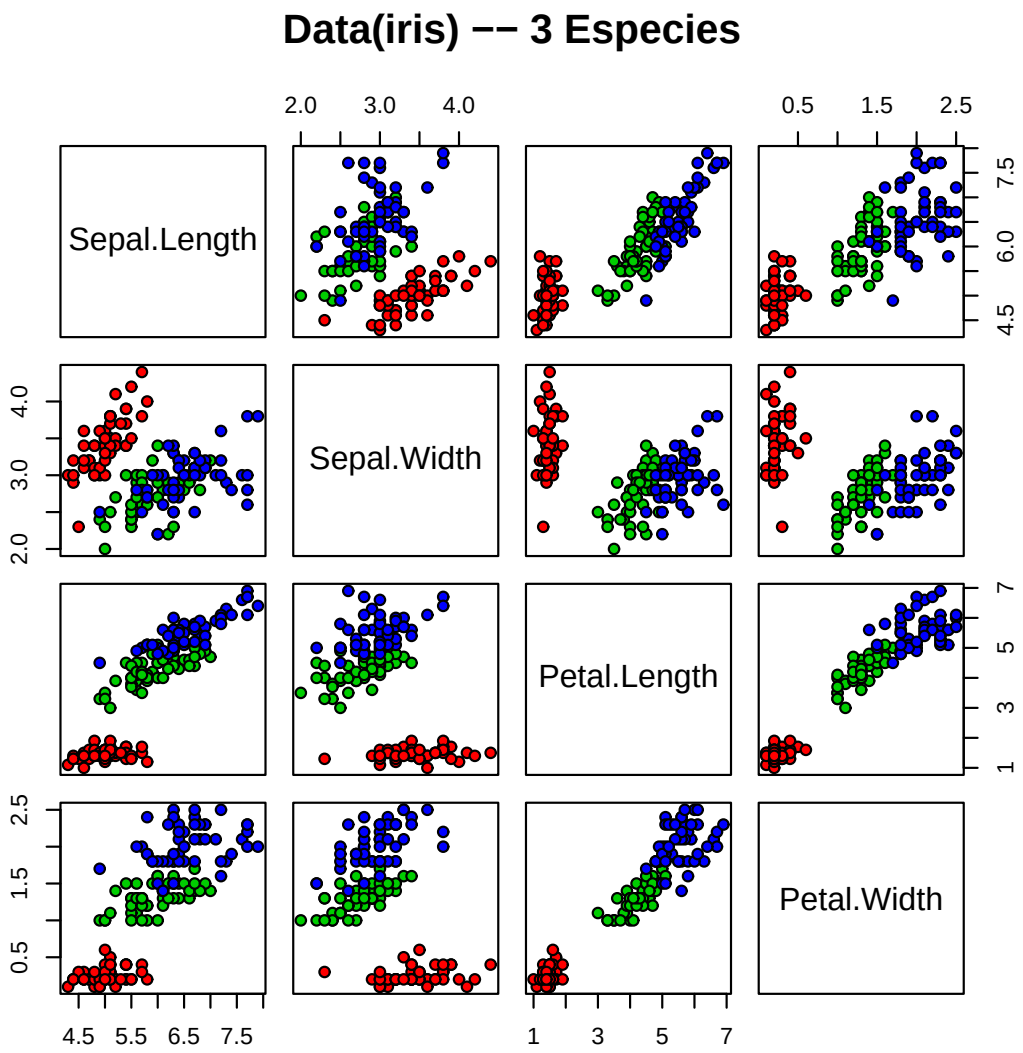

Figura 3.6: *El argumento bg= c("red", "green3", "blue")[codes(iris$Species)] especifica un color diferente para los puntos según el factor especie.*

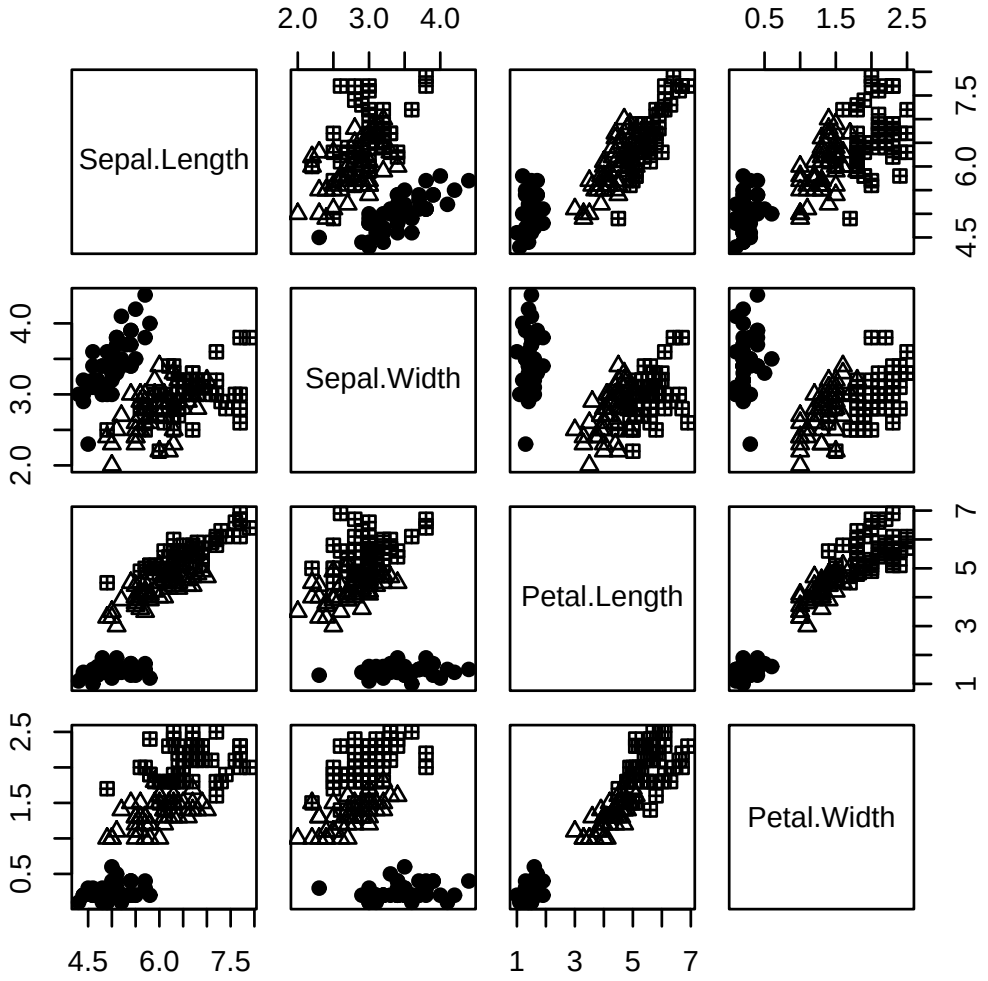

Figura 3.7: *El argumento pch=temp especifica un símbolo diferente para los puntos según el factor especie.*

3.3 Gráficos de independencia

El gráfico chi-plot constituye un método gráfico asociado a una prueba no paramétrica para probar independencia entre un par de variables continuas, desarrollado por Fisher y Switzer (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 (Fisher y Switzer (2001)). 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 periódos son incorrelacionadas.

El gráfico se construye a partir de lo siguiente: Sean $(x_1, y_1), \ldots, (x_n, y_n)$ 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):

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

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

$$G_i = \sum_{j \neq i} I(y_j \leq y_i)/(n-1),$$

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

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

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

El scatterplot de los pares (λ_i, χ_i), $|\lambda_i| < 4\left\{\frac{1}{n-1} - \frac{1}{2}\right\}^2$. λ_i indica la posición del punto (x_i, y_i) respecto al centro de los datos. Adicionalmente, se trazan las líneas horizontales en $\chi = -c_p/n^{\frac{1}{2}}$ y $\chi = c_p/n^{\frac{1}{2}}$, donde c_p se selecciona de forma que aproximadamente $100p\%$ de los pares (λ_i, χ_i) estén entre estas líneas Fisher y Switzer (2001). para $p = 0.90$, 0.95, y 0.99, $c_p =$ 1.54,1.78, y 2.18 respectivamente.

Veamos ahora una implementación en R para un conjunto de datos simulados de una normal bivariada:

```
bivariada<-function(n,mu.x,mu.y,sigma.x,sigma.y,ro){
x<-c(rnorm(n,mu.x,sigma.x))
mu.y.x<-mu.y+ro*sigma.y*(x-mu.x)/sigma.x
sigma.y.x<-sigma.y*sqrt(1-ro^2)
y<-c(rnorm(n,mu.y.x,sigma.y.x))
datos<-cbind(x,y)
datos
}

chi.plot<-function(datos,ro){
n<-nrow(datos)
x<-datos[,1]
y<-datos[,2]
H<-c(rep(0,n))
F<-c(rep(0,n))
G<-c(rep(0,n))
S<-c(rep(0,n))
chi<-c(rep(0,n))
lambda<-c(rep(0,n))

for(i in 1:n){
k1<-0
k2<-0
k3<-0
for(j in 1:n){
if(j!=i){

k1<-ifelse((x[j]<=x[i] & y[j]<=y[i]),k1+1,k1)
k2<-ifelse((x[j]<=x[i]),k2+1,k2)
k3<-ifelse((y[j]<=y[i]),k3+1,k3)
}
}
H[i]<-k1/(n-1)
F[i]<-k2/(n-1)
```

```
G[i]<-k3/(n-1)
S[i]<-sign((F[i]-0.5)*(G[i]-0.5))
c<-c((F[i]-0.5)^2,(G[i]-0.5)^2)
lambda[i]<-4*S[i]*max(c)
chi[i]<-(H[i]-F[i]*G[i])/sqrt(F[i]*(1-F[i])*G[i]*(1-G[i]))
}

nf <- layout(cbind(c(0,1,1,0), c(0,2,2,0)))

plot(x,y,pch=19,main='Scatterplot')
plot(lambda,chi,pch=19,xlim=c(-1,1),ylim=c(-1,1),
main='Chi-plot, p=0.95',sub=paste('ro=',ro),xlab=expression(lambda),
ylab=expression(chi),las=2)
abline(h=0,lty=2)
abline(v=0,lty=2)
abline(h=-1.78/sqrt(n),lty=2)
abline(h=1.78/sqrt(n),lty=2)
}
datos<-bivariada(30,3,5,1,2,0.99)
chi.plot(datos,0.99)
datos<-bivariada(30,3,5,1,2,0.1)
chi.plot(datos,0.1)
datos<-bivariada(30,3,5,1,2,0.01)
chi.plot(datos,0.01)
```

Si se tienen n observaciones de p variables, resulta interesante construir un gráfico que permite visualizar simultáneamente las relaciones entre los pares de variables. Esto es posible mediante el uso de la función "pairs" ya vista, en la cual a su vez se utilice una función que grafique el chi-plot, para uno de los paneles (upper.panel o lower.panel). Un posible programa puede ser el siguiente:

```
panel.chiplot<-function(x,y,...){
usr <- par("usr"); on.exit(par(usr))
       par(usr = c(-1,1,-1, 1))
       par(xaxt='n',yaxt='n')#elimina anotacion en ejes
```

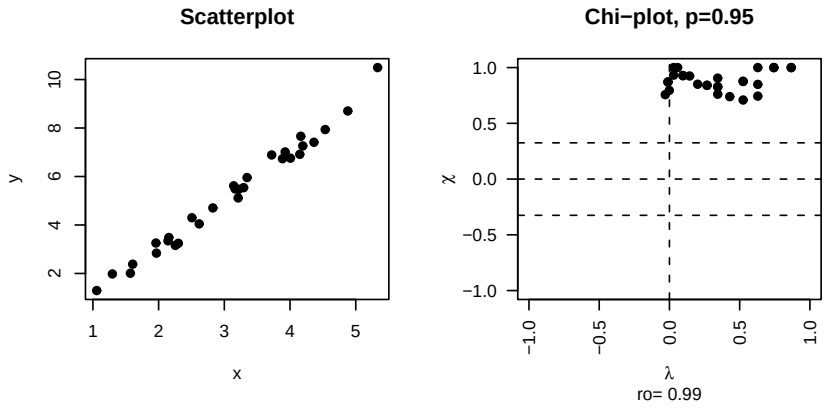

Figura 3.8: *En esta figura aparecen el gráfico de dispersión y chi-plot para los 30 datos simulados de una normal bivariada con* $\rho = 0.99$. *Observe que en el chi-plot los puntos yacen en la parte superior derecha, lo cual es típico cuando existe una relación monótona altamente positiva.*

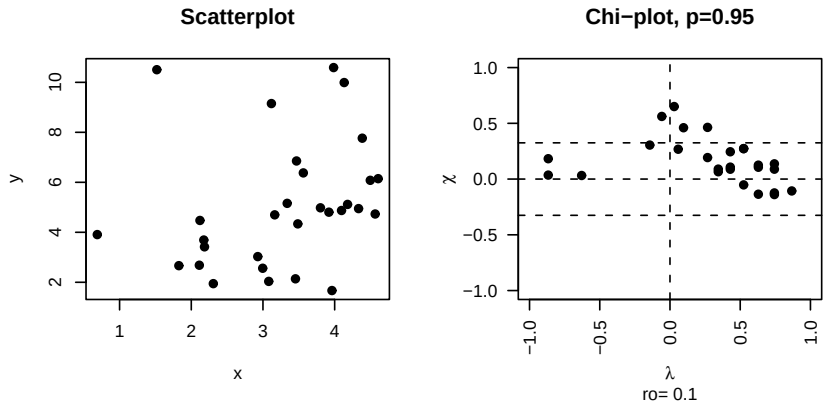

Figura 3.9: *En esta figura aparecen el gráfico de dispersión y chi-plot para los 30 datos simulados de una normal bivariada con* $\rho = 0.1$. *Observe que en el chi-plot los puntos yacen casi por completo entre las dos líneas horizontales del centro. Esto es propio del caso de independencia.*

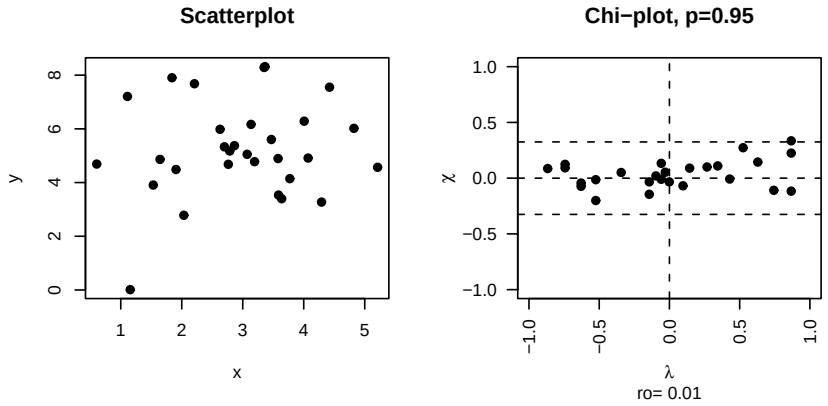

Figura 3.10: *En esta figura aparecen el gráfico de dispersión y chi-plot para los 30 datos simulados de una normal bivariada con* $\rho = 0.01$. *Observe como en el caso anterior que en el chi-plot los puntos yacen casi por completo entre las dos líneas horizontales del centro.*

```
n<-length(x)
H<-c(rep(0,n))
F<-c(rep(0,n))
G<-c(rep(0,n))
S<-c(rep(0,n))
chi<-c(rep(0,n))
lambda<-c(rep(0,n))
for(i in 1:n){
k1<-0
k2<-0
k3<-0
for(j in 1:n){
if(j!=i){
k1<-ifelse((x[j]<=x[i] & y[j]<=y[i]),k1+1,k1)
k2<-ifelse((x[j]<=x[i]),k2+1,k2)
k3<-ifelse((y[j]<=y[i]),k3+1,k3)
}
}
H[i]<-k1/(n-1)
F[i]<-k2/(n-1)
G[i]<-k3/(n-1)
S[i]<-sign((F[i]-0.5)*(G[i]-0.5))
c<-c((F[i]-0.5)^2,(G[i]-0.5)^2)
lambda[i]<-4*S[i]*max(c)
chi[i]<-(H[i]-F[i]*G[i])/sqrt(F[i]*(1-F[i])*G[i]*(1-G[i]))
}
points(lambda,chi,pch=19,...)
abline(h=0,lty=2)
abline(v=0,lty=2)
abline(h=-1.78/sqrt(n),lty=2)
abline(h=1.78/sqrt(n),lty=2)
}
```

Observe que en esta función no se hace uso de “plot” sino de “points”, ya que cuando se invoque “pairs”, los puntos se adicionan a la ventana gráfica ya abierta y no se incurre en el error de crear un nuevo gráfico. También se requerirá la siguiente función, para realizar los gráficos de dispersión por pares:

```
panel.scatter<-function(x,y,...){
par(xaxt='n',yaxt='n')#Elimina anotacion en ejes
points(x,y,pch=19,...)
}
```

Finalmente, obtenemos el gráfico combinado de dispersión y chi - plot, para una matriz de datos $\mathbf{X}$, así :

```
par(xaxt='n',yaxt='n')
pairs(X,lower.panel=panel.chiplot,upper.panel=panel.scatter)
```

Ejemplo:

```
data(iris)
par(xaxt='n',yaxt='n')
pairs(iris[1:4],lower.panel=panel.chiplot,upper.panel=panel.scatter)
par(oma=c(1,1,1,1),new=T,font=2,cex=0.5)
mtext(outer=T,"Ejemplo Matriz de Dispersión con
chi-plot",side=3)
```

3.4 Otros gráficos

3.4.1 Curvas de Andrews

Un gráfico de Andrews está basado en una 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 representado por una componente individual en la suma de la transformada de Fourier. Tradicionalmente, t varía entre $-\pi$ y π para permitir una adecuada representación de los datos. La magnitud de cada variable de un sujeto particular afecta la frecuencia, la

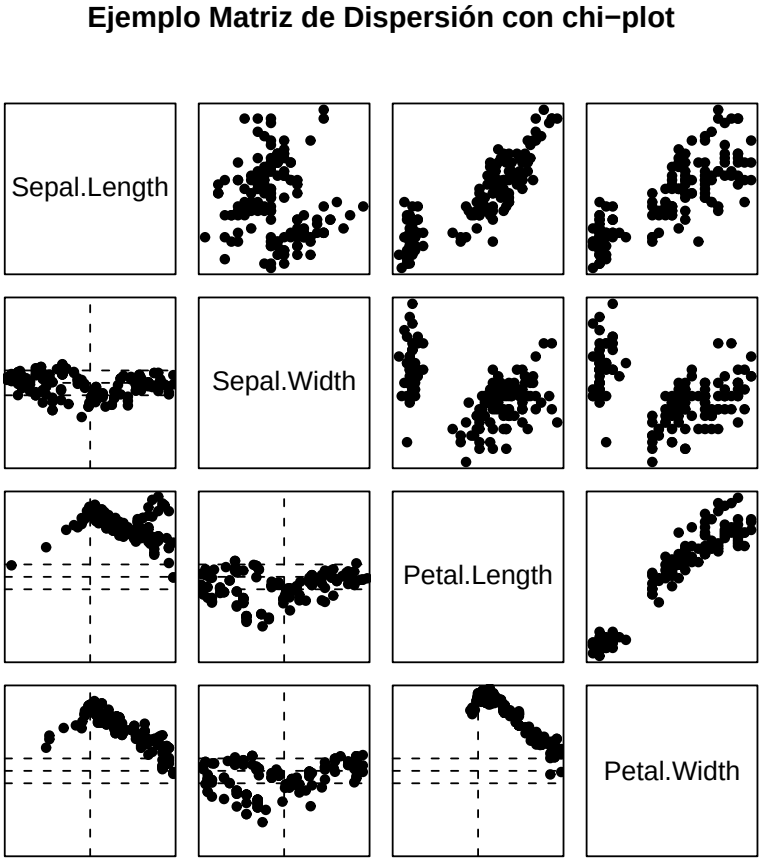

Figura 3.11: *En esta matriz de dispersión aparecen abajo de la diagonal principal los gráficos chi - plot con* $p = 0.95$*, y los respectivos gráficos de dispersión arriba de la diagonal superior, para los datos disponibles en data(iris).*

amplitud y la periodicidad de f, dando una representación única para cada sujeto.

El siguiente ejemplo utiliza el algoritmo sugerido para la identificación de outliers multivariados, por Gnanadesikan (Barnett V., y Lewis T. (1998)), en el cual se elige un grid de valores de t sobre $(-\pi, \pi)$, determina $f_{x_j}(t)$, $j = 1, 2, \ldots, l)$, para cada t se determina los cuantiles 10,25,50,75, y 70 de $f_x(t)$, estos son graficados junto con cualquier valor de $f_{x_j}(t)$ por fuera de los deciles:

```
datos<-matrix(scan('a:/dedos.dat'),ncol=6,byrow=T)
X<-datos[,1:5]
l<-100
n<-nrow(X)
O<-matrix(rep(NA,4*l),ncol=4)
Q<-matrix(rep(0,5*l),ncol=5)
t<-matrix(seq(-pi,pi,len=l),ncol=1)
f.x.t<-matrix(rep(0,n*l),ncol=n)

for(i in 1:l){ for(j in 1:n){
f.x.t[i,j]<-X[j,1]/sqrt(2)+X[j,2]*sin(t[i,1])+X[j,3]*cos(t[i,1])+
X[j,4]*sin(2*t[i,1])+X[j,5]*cos(2*t[i,1])
}
Q[i,]<-quantile(f.x.t[i,],probs=c(0.1,0.25,0.5,0.75,0.9))
o<-order(f.x.t[i,])
f.x.t.o<-f.x.t[i,][o]
O[i,1][f.x.t.o[1]<Q[i,1]]<-f.x.t.o[1]
O[i,2][f.x.t.o[n]>Q[i,5]]<-f.x.t.o[n]

#Observaciones por fuera de deciles:
O[i,3][f.x.t.o[1]<Q[i,1]]<-o[1]
O[i,4][f.x.t.o[n]>Q[i,5]]<-o[n]
}

Aux<-cbind(Q,O[,1:2])
temp1<-as.character(O[,3])
temp2<-as.character(O[,4])
matplot(t,Aux[,1:5],type='o',pch=c("T","Q","M","Q","T"),
```

```
col="black",xlim=c(-pi,pi),ylim=c(2,35),xaxt='n',
yaxt='n',ylab="",cex=0.5)
axis(1,at=c(-pi,-pi/2,0,pi/2,pi),
lab=expression(-pi,-pi/2,0,pi/2,pi))
axis(2,at=c(2,10,16,20,25,30,35),
lab=as.character(c(2,10,16,20,25,30,35)))

for(i in 1:l){
points(t[i,1],Aux[i,6],pch=temp1[i],cex=0.5)
points(t[i,1],Aux[i,7],pch=temp2[i],cex=0.5)
}

title(main='CURVAS DE ANDREWS\nDATOS
MEDIDAS DE LOS PIES DE 10 ESTUDIANTES',cex.main=0.8)
```

Embrechts y Herzberg (1991) presentan variaciones a los gráficos de Andrews que utilizan polinomios de Chebychev y polinomios de Legendre.

3.4.2 Gráfico de Estrellas (stars plots)

Supóngase un conjunto de datos multivariados ordenados matricialmente, de manera que las filas corresponden a las observaciones y p columnas, una por cada variable. En dos dimensiones se pueden construir círculos (uno por cada observación multivariada) de un radio prefijado, con p rayos igualmente espaciados emanando del centro de cada círculo. 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 sus similitudes. Es conveniente estandarizar las observaciones, caso en el cual pueden resultar valores negativos. (Johnson, Wichern (1998)).

En R la función correspondiente es:

```
stars(x, full = TRUE, scale = TRUE, radius = TRUE,
labels = dimnames(x)[[1]], locations =
NULL, xlim = NULL, ylim = NULL, len = 1, colors = NULL,
```

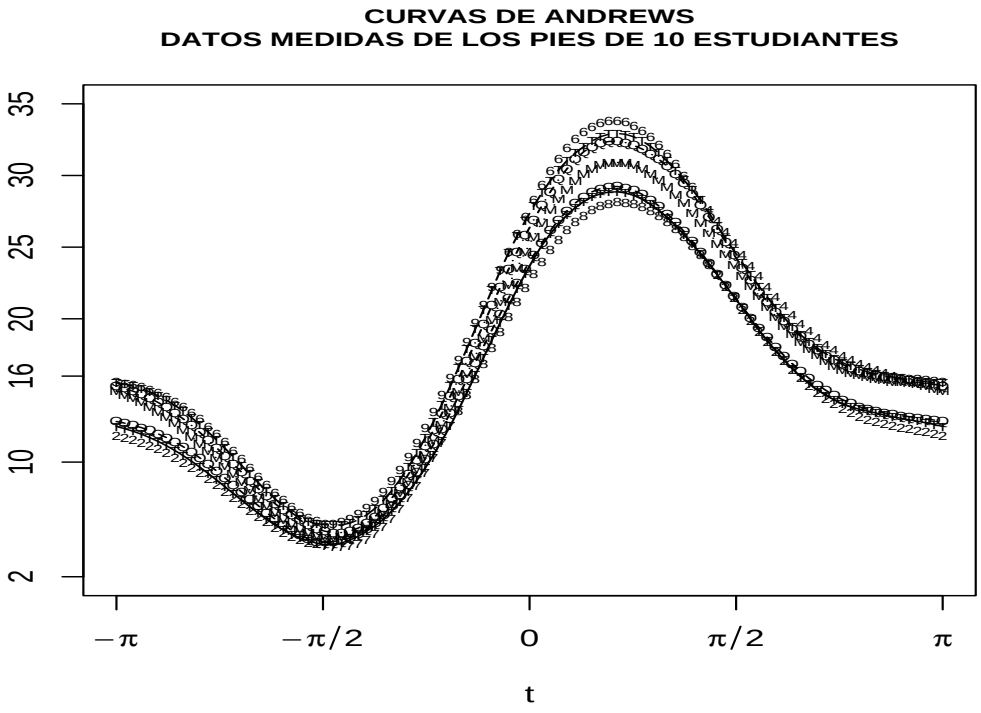

Figura 3.12: *En este gráfico se señalan a las observaciones 2, 3, 4, 6, 7, 8, y 9 por fuera de los deciles que corresponden a los puntos representados con T*

```
key.loc = NULL, key.labels =
NULL, key.xpd = TRUE, draw.segments = FALSE, axes = FALSE,
cex = 0.8, lwd = 0.25, ...)
```

Los argumentos son:

- x: La matriz de datos. Un gráfico de estrella será producido por cada fila de "x". Los valores faltantes serán tratados como si fueran cero.
- full: Argumento lógico: Si es "TRUE", Los gráficos ocuparán un círculo completo. De lo contrario sólo ocuparán un semicírculo superior.
- scale: Argumento lógico: Si es "TRUE", las columnas de la matriz de datos serán escaladas independientemente, de modo que el valor máximo en cada columna toma el valor de 1 y el mínimo de 0. Si es "FALSE", la presunción es que los datos ya han sido escalados por algún otro algoritmo al rango $[0, 1]$.
- radius: Argumento lógico: Si es "TRUE", dibujará el radio correspondiente a cada variable.
- labels: Un vector de caracteres para etiquetar cada gráfico. Por defecto se tiene especificado que use los nombres de las filas de la matriz de datos.
- locations: Corresponde a una matriz de dos columnas con las coordenadas x e y usadas para ubicar cada gráfico de estrella. Si locations es "NULL", es decir, no es pre especificado, los gráficos son colocados en un arreglo rectangular.
- xlim: Es un vector para especificar el rango de las coordenadas x a graficar.
- ylim: Es un vector con el rango de coordenadas y a graficar.
- len: Factor de escala para la longitud del radio de los segmentos.
- colors: Vector de valores enteros, cada uno de los cuales especifica un color para uno de los segmentos. Es ignorado si "draw.segments = FALSE"

- key.loc: Es un vector con las coordenadas x e y para ubicar la unidad que da las claves que identifican las variables en cada start plot.
- key.labels: Es un vector de caracteres para etiquetar los segmentos de la unidad de claves que identifica las variables en cada start plot. Si se omite, la segunda componente de dimnames(x) es usada, si está disponible.
- key.xpd: Anexar un conmutador para la unidad de claves (dibujar y etiquetar), ver "par("xpd")".
- draw.segments: Argumento lógico. Si es "TRUE" dibuja un diagrama de segmento.
- axes: Argumento lógico: Si es "TRUE" se agregan ejes al gráfico.
- cex: Factor de expanside caracter para las etiquetas.
- lwd: Ancho de linea a ser usado para dibujar.
- ...: Argumentos adicionales, pasadas a la primera llamada de "plot()" .

Autor: Thomas S. Dye

Ejemplos:

```
datos.Hombres<-matrix(scan('a:/Acoplah.dat'),ncol=12,byrow=T)
datos.Mujeres<-matrix(scan('a:/Acoplam.dat'),ncol=12,byrow=T)

datos<-rbind(datos.Hombres,datos.Mujeres)

colnames(datos.Hombres)<-c("Sexo","Masa
Corp","Per muslo medio","Anch deltoideo","Per Abdom",
"Per Cadera","Anch
bideltoideo","Estatura","Alt Acromial Parado",
"Pliegue Cutaneo Sub","Pliegue Cutaneo
Trip","edad")

colnames(datos.Mujeres)<-c("Sexo","Masa Corp","Per muslo medio",
```

```
"Anch
deltoideo","Per Abdom","Per Cadera","Anch bideltoideo",
"Estatura","Alt Acromial
Parado","Pliegue Cutaneo Sub","Pliegue Cutaneo Trip","edad")

nombres1<-c(paste("hombre",1:100,sep=""))
nombres2<-c(paste("mujer",1:100,sep=""))
rownames(datos.Hombres)<-nombres1
rownames(datos.Mujeres)<-nombres2

stars(datos.Hombres[1:12,2:12],key.loc=c(11,3),key.labels =
abbreviate(colnames(datos.Hombres)[2:12]),
main='Stars Plots-Datos antropom\'etricos
(hombres)\nPoblaci\'on laboral Colombiana')

stars(datos.Mujeres[1:12,2:12],key.loc=c(11,3),key.labels =
abbreviate(colnames(datos.Mujeres)[2:12]),
main='Stars Plots-Datos antropom\'etricos
(mujeres)\nPoblaci\'on laboral Colombiana')

stars(datos.Hombres[1:12,2:12],key.loc=c(11,3),key.labels =
abbreviate(colnames(datos.Hombres)[2:12]),
main='Stars Plots-Datos antropom\'etricos
(hombres)\nPoblaci\'on laboral Colombiana',
draw.segments=TRUE,len=0.8)
```

Nota: El argumento $key.labels = abbreviate(colnames(datos.Hombres)[2:12])$, indica utilizar en el diagrama de claves los nombres de las columnas 2 a 12 de la matriz datos.hombres. pero abreviándolos, de lo contrario las claves aparecerían completas, pero por su extensión esto resultaría inadecuado. Algo similar puede hacerse con las etiquetas de las observaciones (nombres de filas de la matriz), pero de la siguiente forma: labels = abbreviate(case.names(datos.Hombres))

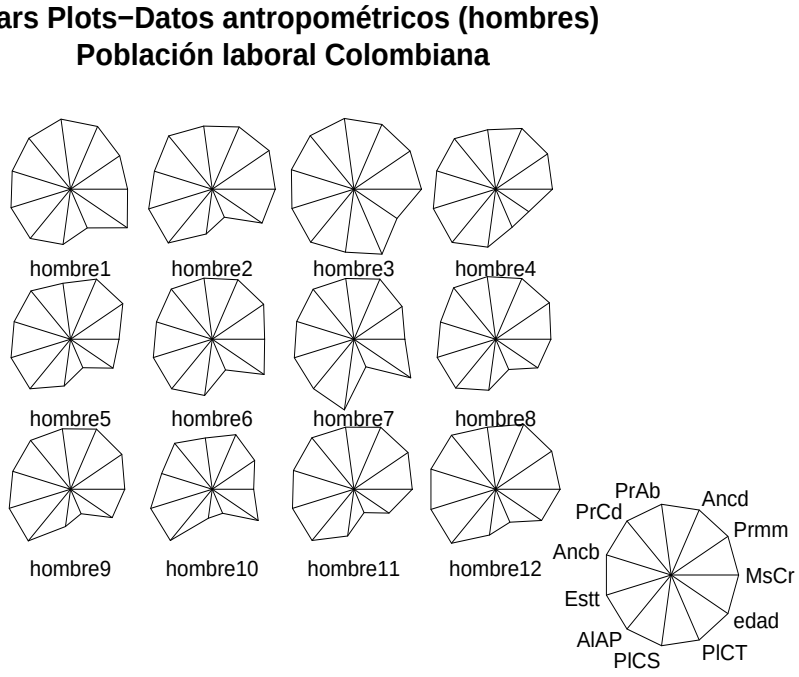

Figura 3.13: *En este gráfico se presentan los primeros 12 datos antropométricos de la base de datos de hombres. Observe que los hombres 3 y 6 presentan un comportamiento que sobresale respecto al resto (el número 3 tiene medidas mayores y el 6 tiene medidas más pequeñas).*

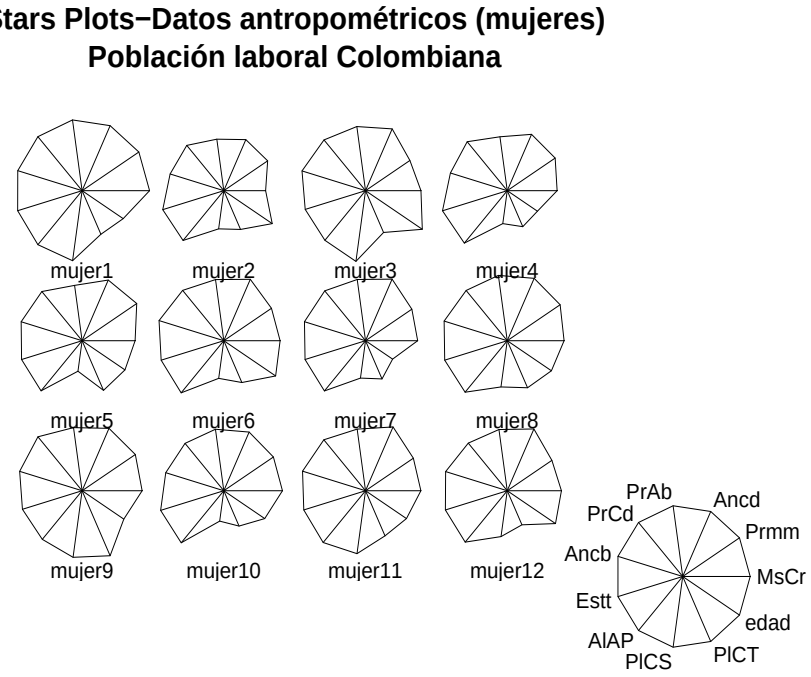

Figura 3.14: *En este gráfico se presentan los primeros 12 datos antropométricos de la base de datos de mujeres. Observe que las mujer 2 y 4 presenta un comportamiento que sobresale respecto al resto (tienen medidas más pequeñas) y que en las variables pliegue cutáneo subescapular (PICS) y en pliegue cutáneo triceps (PICT) parece mayor variabilidad (en la parte inferior de las estrellas observe cómo varíel patrón).*

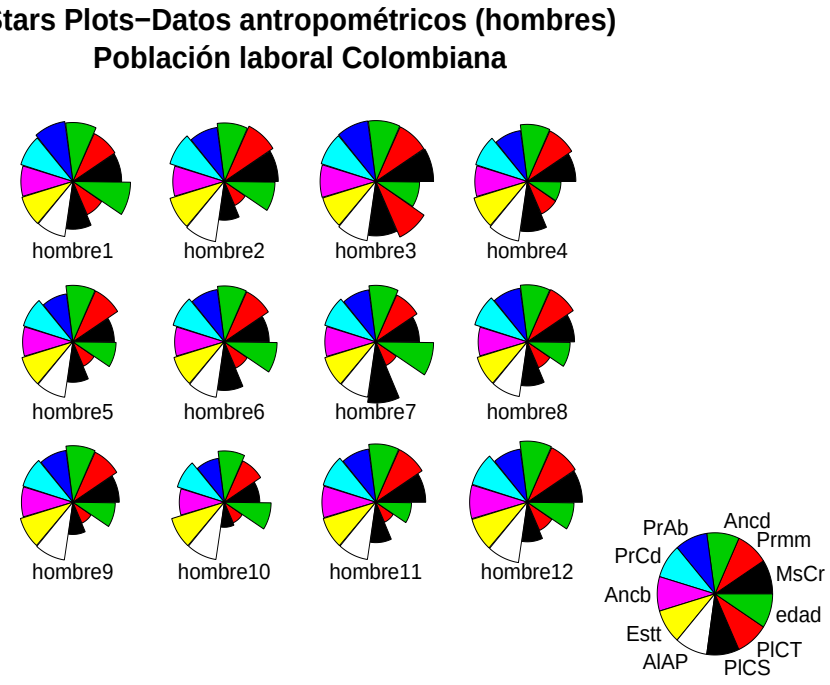

Figura 3.15: *En este gráfico se presentan los mismos primeros 12 datos antropométricos de la base de datos de hombres, pero con una modificación, los gráficos presentan sectores circulares de radios proporcionales a los valores de cada variable en el respectivo sujeto, además que cada sector ha sido coloreado. Esto fue posible especificando el argumento draw.segments=TRUE.*

Capítulo 4

Gráficos para Modelos Estadísticos en R

4.1 Regresión

4.1.1 Gráficos en Regresión Lineal Simple

En regresión simple es posible obtener de manera sencilla los gráficos de dispersión, lí mites de confianza, recta ajustada y de residuales, como se ilustra en el siguiente ejemplo:

```
datos<-matrix(scan('a:/r9.txt'),ncol=2,byrow=F) datos
      [,1] [,2]
 [1,]   85  5.2
 [2,]   86  6.5
 [3,]   87  7.2
 [4,]   94 10.2
 [5,]   97 13.8
 [6,]   96 11.2
 [7,]   86  6.6
 [8,]   90  9.0
 [9,]   83  6.3
[10,]   85  6.2
[11,]   87  6.8
[12,]   94 11.0
```

```
[13,]   89  8.0

anos<-datos[,1]
precio<-datos[,2]
reg.pr.ano<-lm(precio~anos)
summary(reg.pr.ano)
Call:
lm(formula = precio ~ anos)

Residuals:
     Min       1Q   Median       3Q      Max
-0.90764 -0.36693 -0.03728  0.12558  1.33666

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -38.91191    3.91815  -9.931 7.92e-07 ***
anos          0.52964    0.04389
12.067 1.10e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7046 on 11 degrees
of freedom Multiple R-Squared: 0.9298,
Adjusted R-squared: 0.9234 F-statistic: 145.6 on 1 and 11 DF,
p-value: 1.099e-007

anova(reg.pr.ano) Analysis of Variance Table

Response: precio
          Df Sum Sq Mean Sq F value    Pr(>F)
anos       1 72.288  72.288   145.6 1.099e-07 ***
Residuals 11  5.461   0.496
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

par(mfrow=c(2,2))
plot(reg.pr.ano)
par(oma=c(1,1,1,1),new=T,font=2,cex=0.5)
```

```
mtext(outer=T,"Gráficos básicos Regresión simple
precio vs. año R9",side=3)

predict(reg.pr.ano,se.fit = TRUE)

#obtiene valores de predicción y error estandar

temp1<- predict(reg.pr.ano,interval="prediction")

temp2<- predict(reg.pr.ano,interval="confidence")

Interv.pred<-temp1[order(anos),]

#ordena las filas según orden ascendente de años

Interv.conf<-temp2[order(anos),]

#ordena las filas según orden ascendente de años

temp3<-anos[order(anos)]

#ordena en forma ascendente los años

temp4<-precio[order(anos)]

x<-matrix(temp3,ncol=1)

#Construimos una matriz que tiene col 1 los valores predichos,
col 2 y 3

#los intervalos de confianza y col 4 y 5 los intervalos
de predicción:

y<-matrix(cbind(Interv.conf,Interv.pred[,-1]),ncol=5)

#Graficamos las columnas de y vs. x

matplot(x,y,lty=c(1,2,2,3,3),xlim=c(82,97),ylim=c(3,14.5),
```

```
type="l",col=c(1,2,2,4,4),
main="Precios predichos, e intervalos de\nconfianza y
predicción del 95%",xlab="años", ylab="precio predicho R9")

#Graficamos los puntos u observaciones originales superpuestos
 al anterior gráfico:

par(new=T)
plot(temp3,temp4,xlim=c(82,97),ylim=c(3,14.5),xlab="",ylab="")

#o bien:

plot(temp3,temp4,xlim=c(82,97),ylim=c(3,14.5),
main="Precios predichos, e intervalos de\nconfianza
y predicción del 95%",xlab="años", ylab="precio predicho R9")
matplot(x,y,add=T,lty=c(1,2,2,3,3),pch=c(1,2,2,3,3),xlim=c(82,97),
ylim=c(3,14.5),type="o",col=c(1,2,2,4,4))
```

La función matplot utilizada aquí 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. El gráfico se obtiene como sigue:

```
matplot(x, y, type = "p", lty = 1:5, lwd = 1, pch = NULL,
col = 1:6, cex = NULL, xlab =
NULL, ylab = NULL, xlim = NULL, ylim = NULL, ...,
add = FALSE, verbose =
getOption("verbose"))
```

Sus Argumentos son:

- x,y: 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, el dado es tomado como y, y se utiliza como x un vector de 1:n. Valores NA son permitidos.
- type: Un vector para indicar el tipo de gráfico para cada columna de y (como se vió con la función plot), por defecto es p.

- lty,lwd: Vectores para indicar los tipos y anchos de líneas para cada columna de y.
- pch: Caracter o vector de caracteres o enteros para definir los "plotting characters", por cada columna de y second, etc. The default is the digits (1 through 9, 0).
- col: Vector de colores, aunque si no se especifica 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 ejes x e y.
- ...: Parámetros gráficos adicionales (ver "par")
- add: Argumento lógico. Si es TRUE, el gráfico es agregado a un gráfico previo. Se debe garantizar que los rangos de los ejes en ambos gráficos sean los mismos.
- verbose: Argumento lógico. Si es "TRUE", escribe una linea de lo que hace la función, por ejemplo:

```
matplot: doing  5  plots with   col= ("1" "2" "2" "4" "4")
pch= ("1" "2" "3" "4" "5")
...
```

4.2 Algunos Ajustes de Curvas por Regresión no Paramétrica

4.2.1 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 funcón que hace la interpolación, veamos:

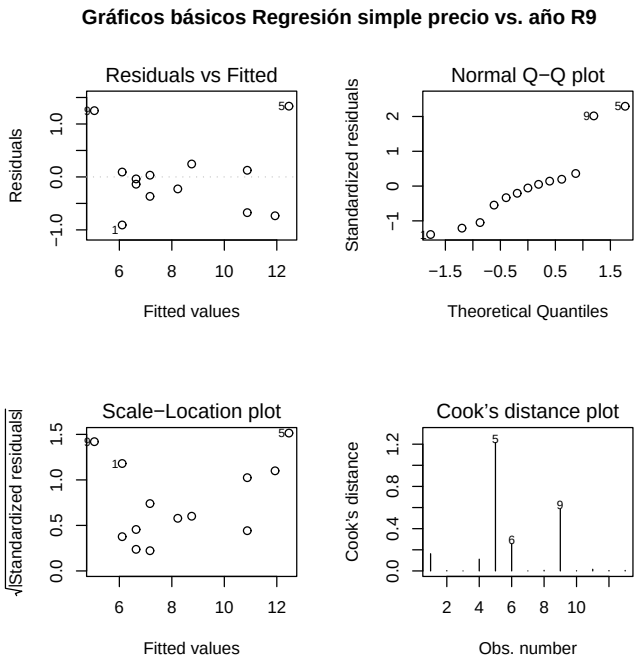

Figura 4.1: *Los gráficos de residuos vs. valores predichos, Q-Q plot de residuales, residuos estandarizados vs. valores predichos y distancias de Cook's*

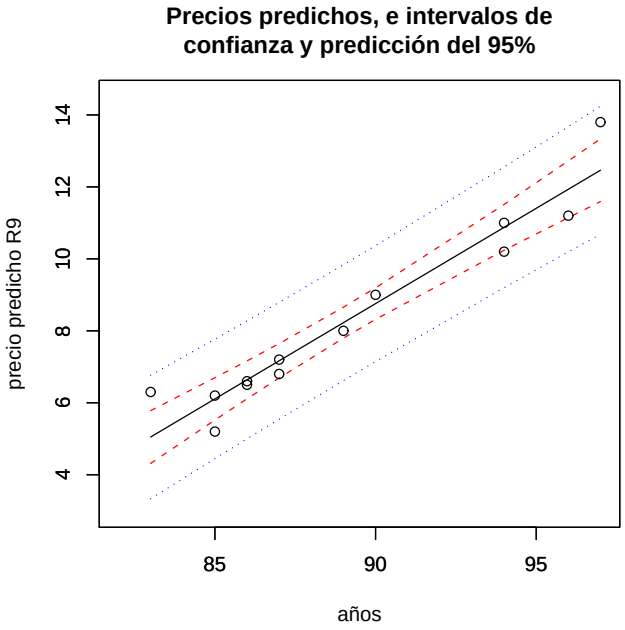

Figura 4.2: *Valores ajustados e intervalos de confianza (bandas internas) y de prediccón (bandas externas).*

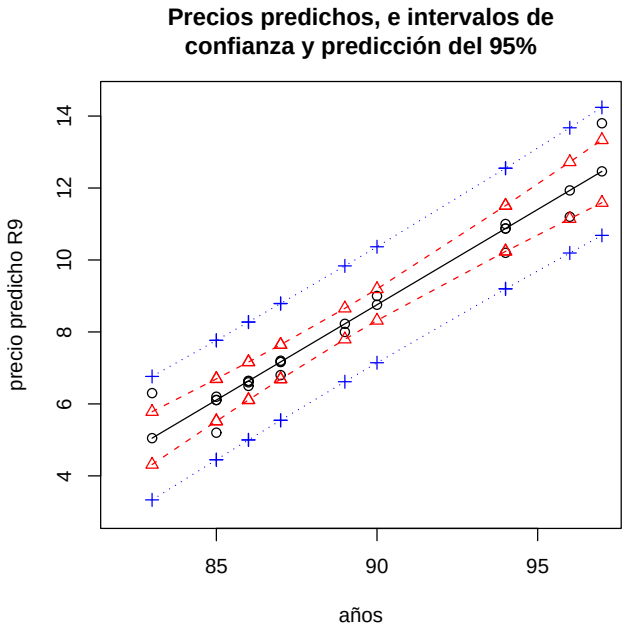

Figura 4.3: *El mismo ejemplo anterior pero con definición de type='o'.*

```
splinefun(x, y, method = "fmm")

spline(x, y, n = 3*length(x), method = "fmm",
xmin = min(x), xmax = max(x))
```

Valor: spline devuelve un lista de componentes x e y que dan las coordenadas en donde se da la interpolación y los valores interpolados.

splinefun devuelve una función la cual realiza interpolación spline cúbica de los puntos dados. Esta es a menudo más útil que spline.

Argumentos:

- x,y: Los vectores que contienen las coordenadas de puntos a ser interpoladas. Alternativamente una estructura de graficación individual puede ser especificada: ver 'xy.coords'.
- method: Especifica el tipo de spline a ser usado. los valores posibles son (entre comillas):
  - "fmm"
  - "natural"
  - "periodic"

  Si method="fmm", se usa el spline de Forsythe, Malcolm y Moler, en el cual se ajusta un spline cúbico exacto a través de los cuatro puntos en cada extremo de los datos, y éste es usado para determinar las condiciones de conclusión. Por defecto method="fmm".
- n: Para indicar que la interpolación ocurre en 'n' puntos igualmente espaciados en el intervalo $[xmin, xmax]$.
- xmin: Punto extremo en el lado izquierdo del intervalo de interpolación.
- xmax: Punto extremo en el lado derecho del intervalo de interpolación.

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.

referencias:

- Forsythe, G. E., Malcolm, M. A. and Moler, C. B. (1977) Computer Methods for Mathematical Computations.

también: 'approx' y 'approxfun' para interpolación constante y lineal. En el paquete 'splines', ver 'interpSpline' y 'periodicSpline'. Este paquete también genera bases spline que pueden ser usados para regresiones spline. En el paquete 'modreg' ver 'smooth.spline' para splines de suavizamiento.

Ejemplo 1:

```
n <- 50
x <- 1:n
y <- rnorm(n)
splinefun(x, y, method = "fmm") #produce lo sgte:

function (x) {
    .C("spline_eval", z$method, length(x), x = as.double(x),
        y = double(length(x)), z$n, z$x, z$y, z$b, z$c, z$d,
        PACKAGE = "base")$y}

<environment: 01C58844>

#Para graficar la curva ajustada:

plot(x, y, main =
paste("spline[fun](.) a través de", n, "puntos"))

lines(spline(x, y))
```

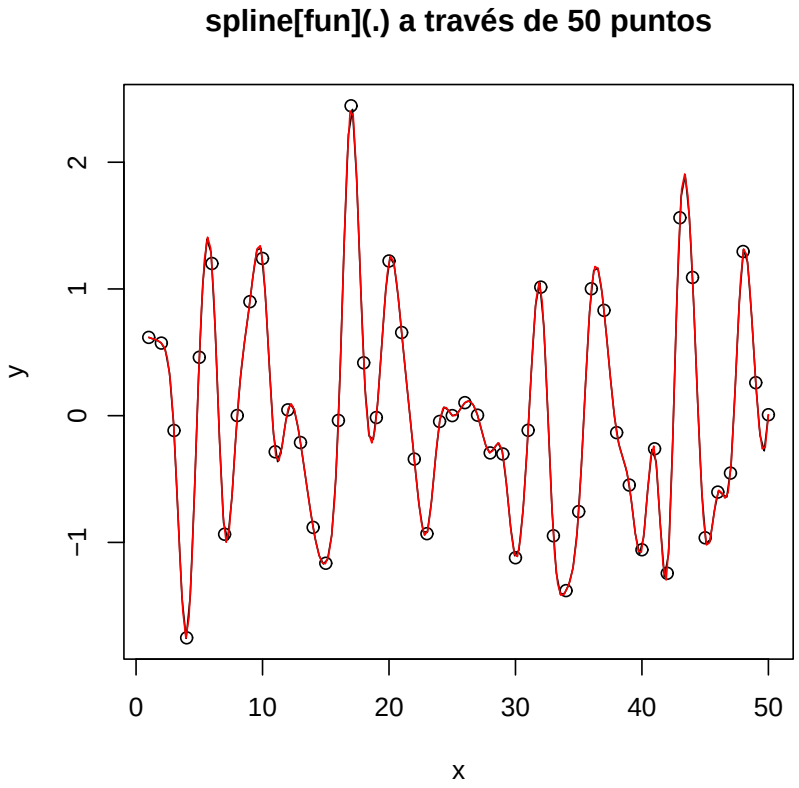

Figura 4.4: *curva ajustada por spline con método fmm.*

```
lines(spline(x, y, n = 201), col = 2)
```

Ejemplo 2: veamos los spline por los tres métodos:

```
y1 <- (x-6)^2

plot(x, y1, main = "spline(.) -- por los 3 métodos")

lines(spline(x, y1, n = 201), lty = 1)
```

```
lines(spline(x, y1, n = 201, method = "natural"),
lty = 2)
lines(spline(x, y1, n = 201, method = "periodic"), lty = 3)

legend(6,1500, c("fmm","natural","periodic"),  lty=1:3)
```

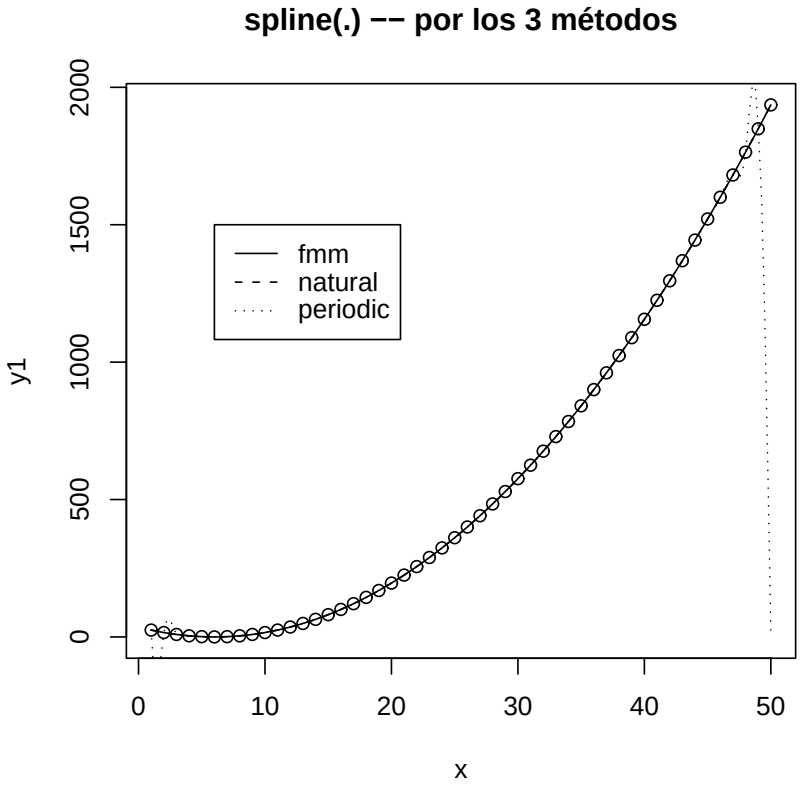

Figura 4.5: *Curvas ajustadas por spline con los métodos fmm, natural y periodic. Vemos que el ajuste con el último método difiere de los dos anteriores y presenta problemas.*

Ejemplo 3: Lo siguiente permite extrapolar con la curva spline ajustada previamente a un conjunto de puntos, en un intervalo dado de la variable x:

```
f <- splinefun(x, y1)
f
function (x) {
    .C("spline_eval", z$method, length(x), x = as.double(x),
        y = double(length(x)), z$n, z$x, z$y, z$b, z$c, z$d,
        PACKAGE = "base")$y
}
<environment: 0178EBC8>


ls(envir = environment(f))
[1] "z"

splinecoef <- eval(expression(z), envir = environment(f))

#'eval' evalua una expresion R en un ambiente especificado

curve(f(x), 1, 10, col = "black", lwd = 1.5,
main="Valores extrapolados\najuste spline")
points(splinecoef, col = "blue", cex = 2)
```

4.2.2 Regresión Kernel

En R existe disponible en el paquete o librería 'modreg' la función 'ksmooth' con la cual es posible obtener la curva ajustada con la función kernel determinado, utilizando el estimador de Nadaraya-Watson:

```
ksmooth(x, y, kernel = c("box", "normal"), bandwidth = 0.5,
range.x = range(x), n.points
= max(100, length(x)), x.points)
```

Valor: Esta función produce una lista con componentes x : los valores en orden creciente, en los cuales es evaluado el ajuste suavizado; y y los valores ajustados correspondientes a x.

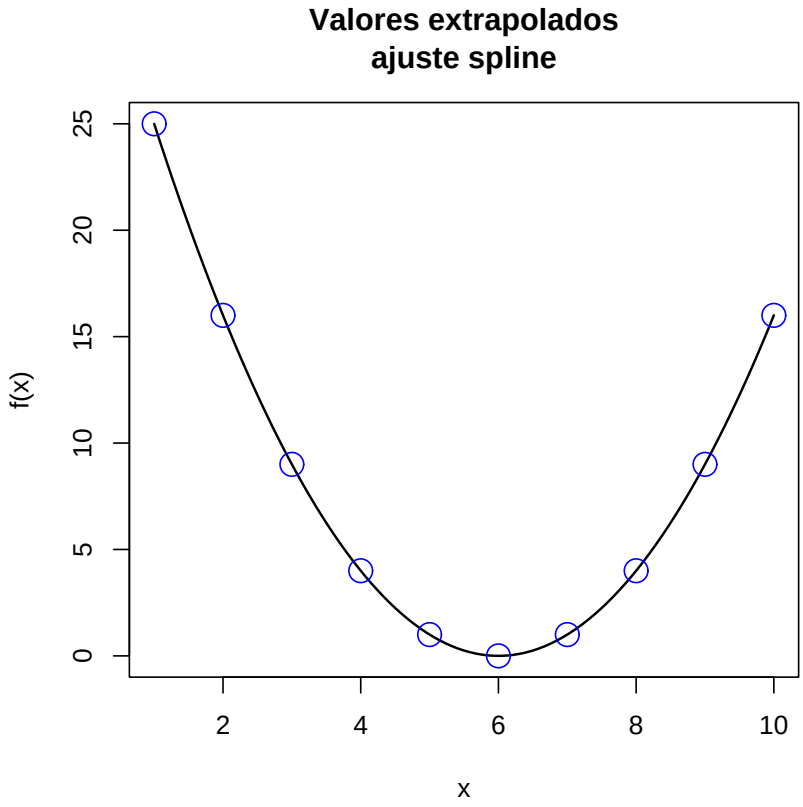

Figura 4.6: *La función 'curve' junto con 'points' han sido utilizadas para graficar los valores extrapolados del spline ajustado previamente, que ha sido asignada a un objeto R donde quedan guardados los coeficientes de la curva ajustada permitiendo evaluar (extrapolar) en un rango de valores x específico.*

Argumentos:

- x: Los valores de la variable x o predictora.
- y: Los valores de la variable y o respuesta.
- kernel: El kernel a ser usado.
- bandwidth: El bandwidth. Los kernels son escalados de modo que sus cuartiles (vistos como densidades de probabilidad) estén en $+/-0.25 \times$ *bandwidth*
- range.x: El rango de puntos a ser cubiertos en la salida (*output*) o resultado.
- n.points: El número de puntos en los cuales se va a evaluar el ajuste.
- x.points: Los puntos en los cuales se evaluará el ajuste suavizado. Si no se especifica, entonces 'n.points' son elegidos uniformemente para cubrir 'range.x'.

Autor: B. D. Ripley

Ejemplo:

```
datos<-matrix(scan('a:/r9.txt'),ncol=2,byrow=F)
anos<-datos[,1]
precio<-datos[,2]
plot(anos,precio,pch=19,cex=1.5,xlab="años",ylab="precios",
main="Ajustes de Precios
Renault 4\npor regresión kernel")

library(modreg)
kernel1<-ksmooth(anos,precio,"normal",bandwidth=2)
kernel2<-ksmooth(anos,precio,"normal",bandwidth=5)
kernel3<-ksmooth(anos,precio,"normal",bandwidth=10)

lines(kernel1,lty=1)
lines(kernel2,lty=2)
lines(kernel3,lty=3)
```

```
legend(84,12,c("bandwidth=2","bandwidth=5","bandwidth=10"),
lty=1:3)
```

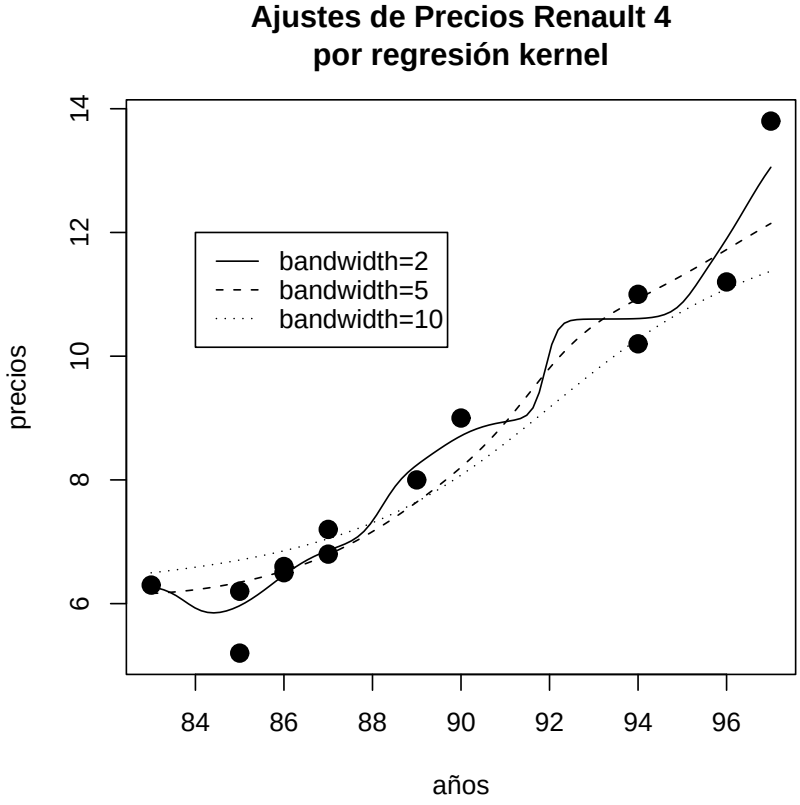

Figura 4.7: *Curvas ajustadas por tres suavizadores kernel. Observe el efecto del argumento 'bandwidth'.*

4.2.3 LOESS

El ajuste de polinomios locales propuesto por Cleveland, puede lograrse en R mediante la función 'loess' del paquete o librería 'modreg':

```
loess(formula, data, weights, subset, na.action,
model = FALSE, span = 0.75, enp.target,
degree = 2, parametric = FALSE, drop.square = FALSE,
normalize = TRUE, family =
c("gaussian", "symmetric"),
method =c("loess", "model.frame"), control =
loess.control(...), ...)
```

Argumentos:

- formula: Una fórmula que especifica la respuesta y uno o más predictores numéricos (mejor especificados vía una interacción, pero también pueden ser especificados aditivamente).
- data: Un marco de datos opcional dentro del cual se busca primero la respuesta, los predictores y los pesos.
- weights: Pesos opcionales para cada caso.
- subset: Una especificación opcional de un subconjunto de los datos a usar.
- na.action: La acción a seguir con valores faltantes en la respuesta o los predictores. Por defecto es parar.
- model: Debe retornarse el marco del modelo?
- span: El parámetro α que controla el grado de suavizamiento.
- enp.target: Una forma alternativa para especificar 'span', como el número aproximado de parámetros a ser usados.
- degree: El grado de los polinomios a ser usados, hasta 2.
- parametric: Debe cualquiera de los términos ser ajustado globalmente en vez de localmente?. Los términos pueden especificarse por nombre, número o como un vector lógico con longitud igual al número de predictores.

- drop.square: Para ajustes con más de un predictor y 'degree=2', debe el término cuadrático y los términos cruzados, ser excluídos para predictores en particular? Los términos se especifican de la misma forma que en 'parametric'.

- normalize: Deben los predictores ser normalizados a una escala común si hay más de uno? La normalización usada consiste en usar la desviación estándar recortada al 10%. Ajustar este argumento a 'FALSE' exige saber que los predictores en coordenadas espaciales y otros están en una escala común.

- family: Si se especifica como "gaussian" (entre comillas) el ajuste se hace por m´cuadrados, y si se especifica como "symetric" (entre comillas) un estimador M redescendiente es usado con una función biponderada de Tukey.

- method: Ajusta el modelo si se especifica como "loess" o sólo extrae el marco del modelo si se especifica como "model.frame".

- control: Parámetros de control: ver 'loess.control'.

- ...: Parámetros de control pueden también proporcionarse directamente.

Detalles:

El ajuste es hecho localmente. Es decir, para el ajuste en un punto x, el ajuste es hecho usando los puntos en un 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 vecindad 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 $(1-(dist/maxdist)^3)^3$. Para $\alpha > 1$, todos los puntos son usados, con la 'distancia máxima' asumida como α^1/p veces la distancia máxima actual para p variables explicativas.

Para la familia por defecto, el ajuste es por mínimo cuadrados (ponderados). para "symetric" son usadas unas pocas iteraciones de un procedimiento de estimación M con biponderación de Tukey. Tener en cuneta que como el valor inicial es el ajuste de mínimos cuadrados, éste no necesita ser un ajuste

muy resistente.

puede ser importante ajustar la lista de control para alcanzar un velocidad aceptable. Ver 'loess.control'.

Autor: B.D. Ripley, based on the 'cloess' package of Cleveland, Grosse and Shyu.

referencias: W.S. Cleveland, E. Grosse and W.M. Shyu (1992) Local regression models. Chapter 8 of Statistical Models in S eds J.M. Chambers and T.J. Hastie, Wadsworth Brooks/Cole.

Ver también: 'loess.control', 'predict.loess'. 'lowess', el ancestro de 'loess'.

Ejemplos:

```
datos<-matrix(scan('a:/r9.txt'),ncol=2,byrow=F)
anos<-datos[,1]
precio<-datos[,2]

library(modreg)
loess.precios1<-loess(precio~anos,family="gaussian",
method="loess")

loess.precios1
Call: loess(formula = precio ~ anos)

Number of Observations: 13
Equivalent Number of Parameters: 4.94
Residual Standard Error:
0.5998
```

```
p<-predict(loess.precios1,se=TRUE)
p

$fit
 [1]  6.005998  6.378010  7.042103 10.520526
 [5] 13.192575 12.149888  6.378010  8.785513
 [9]  6.190112  6.005998  7.042103 10.520526
[13]  8.182549

$se.fit
 [1] 0.2852881 0.2666417 0.3401038 0.3660114
 [5] 0.5128947 0.3344645 0.2666417 0.4386076
 [9] 0.5813140 0.2852881 0.3401038 0.3660114
[13] 0.3599517

$residual.scale
[1] 0.5998412

$df
[1] 7.325173

#Para extrapolación:

loess.precios2<-loess(precio~anos,control =
loess.control(surface = "direct"))
predict(loess.precios2,data.frame(anos=c(98,99,100)),se=TRUE)

$fit
[1] 14.40041 15.78864 17.36360

$se.fit
[1] 0.8313466 1.2559199 1.7740425

$residual.scale
[1] 0.5999703

$df
```

```
[1] 7.325593
```

Si se requiere 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 un lista de valores x e y

```
scatter.smooth(x, y, span = 2/3, degree = 1,
family = c("symmetric", "gaussian"), xlab =
deparse(substitute(x)), ylab = deparse(substitute(y)),
ylim = range(y, prediction$y),
evaluation = 50, ...) loess.smooth(x, y, span = 2/3,
degree = 1, family = c("symmetric",
"gaussian"), evaluation=50, ...)
```

Argumentos:

- x: Coordenadas x para el gráfico de dispersión.
- y: Coordenadas y para el gráfico de dispersión.
- span: Parámetro de suavizamiento para 'loess'.
- degree: Grado de polinomio local usado.
- family: Si se especifica como "gaussian" (entre comillas) el ajuste es por mínimos cuadrados; si se especifica como "symetric" (entre comillas) es usado un estimador M redescendiente.
- xlab: Etiqueta para el eje x.
- ylab: Etiqueta para el eje y.
- ylim: El límite y del gráfico.
- evaluation: El número de puntos en los cuales se evalúa la curva suavizada
- ...: Parámetros gráficos.

Detalles: 'loess.smooth' es una función auxiliar.

Autor: B.D. Ripley

Ejemplo:

```
library(modreg)
datos<-matrix(scan('a:/r9.txt'),ncol=2,byrow=F)
anos<-datos[,1]
precio<-datos[,2]

layout(matrix(c(1,2,3,4),ncol=2,nrow=2,byrow=T),
widths=c(7,7),heights=c(6,6),respect=T)

par(mar=c(4,4,2,1))
scatter.smooth(anos,precio,span=2/3,family="gaussian",
xlab="años",ylab="precio",
main=expression(paste(alpha==2/3,sep="
",'family="gaussian"')),pch=19,lwd=2,cex.main=0.8,
cex.lab=0.8,cex.axis=0.8,cex.sub=0.8)

par(mar=c(4,4,2,1))
scatter.smooth(anos,precio,span=2/3,family="symmetric",
xlab="años",ylab="precio",
main=expression(paste(alpha==2/3,sep="
",'family="symmetric"')),pch=19,lwd=2,cex.main=0.8,
cex.lab=0.8,cex.axis=0.8,cex.sub=0.8)

par(mar=c(4,4,2,1))
scatter.smooth(anos,precio,span=0.8,family="gaussian",
xlab="años",ylab="precio",
main=expression(paste(alpha==0.8,sep="
",'family="gaussian"')),pch=19,lwd=2,cex.main=0.8
,cex.lab=0.8,cex.axis=0.8,cex.sub=0.8)

par(mar=c(4,4,2,1))
```

```
scatter.smooth(anos,precio,span=0.8,family="symmetric",
xlab="años",ylab="precio",
main=expression(paste(alpha==0.8,sep="
",'family="symmetric"')),pch=19,lwd=2,cex.main=0.8,
cex.lab=0.8,cex.axis=0.8,cex.sub=0.8)

par(oma=c(1,1,1,1),font=2)
mtext(outer=T,"Ajuste LOESS de precios Renault
4",side=3,cex=1)
```

Alternativamente, también es posible graficar la curva ajustada por LOESS junto con el scatter plot utilizando la función 'loess.smooth', y con lo cual se pueden superponer diferentes curvas ajustadas, de la siguiente forma:

```
plot(anos,precio,xlab="años",ylab="precio",
main="Ajuste LOESS\nPrecios Renault
4",pch=19,lwd=2,font.main=2)
lines(loess.smooth(anos,precio,span=2/3,family="symmetric"),
lty=1,lwd=2)
lines(loess.smooth(anos,precio,span=0.8,family="symmetric"),
lty=3,lwd=2)
lines(loess.smooth(anos,precio,span=2/3,family="gaussian"),
lty=5,lwd=2)
lines(loess.smooth(anos,precio,span=0.8,family="gaussian"),
lty=4,lwd=2)
legend(83,14,c("span=2/3,family=symmetric",
"span=0.8,family=symmetric","span=2/3,family=gaussian",
"span=0.8,family=gaussian"
),lty=c(1,3,5,4),lwd=2,cex=0.95)
```

4.3 Análisis de Componentes Principales

Los gráficos bsicos en el análisis de componentes principales son:

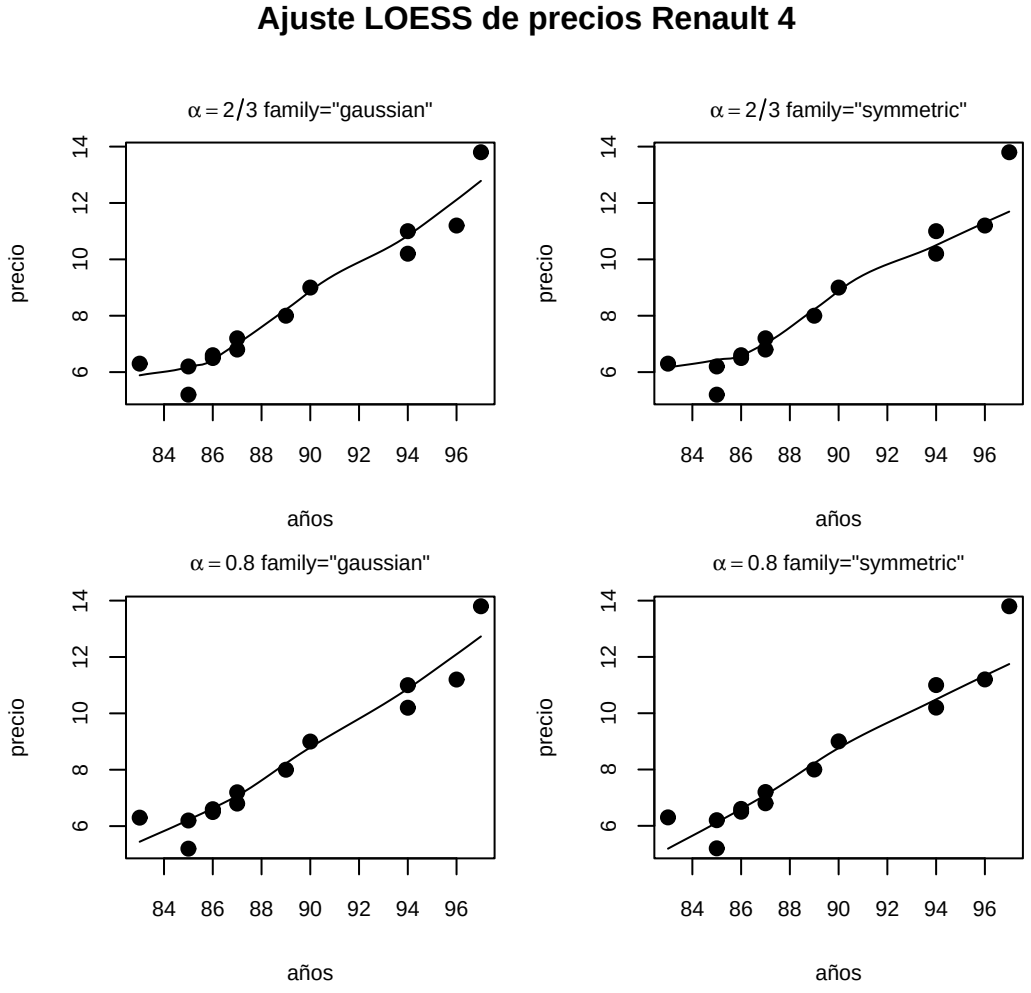

Figura 4.8: *Curvas ajustadas por loess. Observe el efecto del argumento 'span' (α) y del argumento 'family'.*

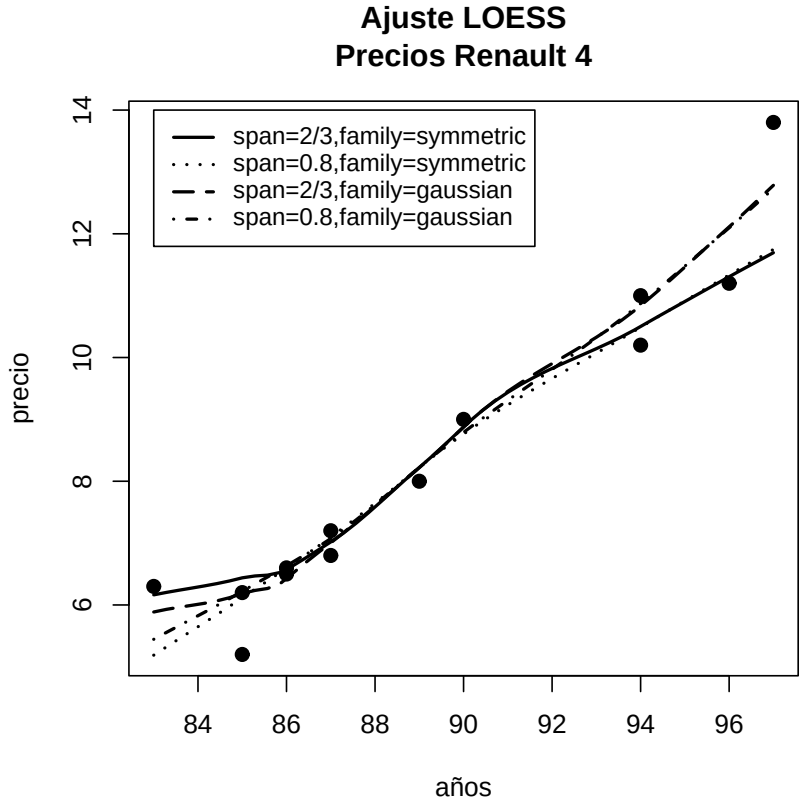

Figura 4.9: *Observe el efecto del valor del parámetro de suavizamiento 'span' y del argumento 'family'. Las curvas ajustadas por "gaussian" son más influenciadas por valores extremos, de modo que estos "tiran" hacia sí la curva ajustada; en cambio, las ajustadas por "symmetric" no son tan influenciadas por esta clase de puntos. ¿cuál de estos ajustes es mejor?.*

- "Scree Plot",
- "Scores Plot", y
- "Principal Components Pattern Plot"

A continuación se presentan dos posibles programas para obtener las componentes principales dada una matriz de datos $\mathbf{X}$ de dimensión $n \times p$, 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.

```
x<-matrix(c(c(16,12,13,11,10,9,8,7,5,3,2,0),
c(8,10,6,2,8,-1,4,6,-3,-1,-3,0)),ncol=2,byrow=F)

x
      [,1] [,2]
 [1,]   16    8
 [2,]   12   10
 [3,]   13    6
 [4,]   11    2
 [5,]   10    8
 [6,]    9   -1
 [7,]    8    4
 [8,]    7    6
 [9,]    5   -3
[10,]    3   -1
[11,]    2   -3
[12,]    0    0

#COMPONENTES PRINCIPALES CON BASE
#EN LA DESCOMPOSICION SVD DE X:
compprinc1<-function(x){
medias.x<-apply(x,2,mean)
medias<-t(medias.x)
uno<-c(rep(1,nrow(x)))
matriz.uno<-matrix(t(uno))
matriz.medias<-matriz.uno%*%medias
datos.centrados<-x-matriz.medias
SIGMA.X<-var(x)
```

```
desc.val.sing<-svd(datos.centrados)
P<--desc.val.sing$u
Q<--desc.val.sing$v
d<-desc.val.sing$d
D<-diag(d)
scores<-P%*%D
lambdas<-d^2/(nrow(x)-1) #valores propios de
                         #la matriz de covarianza

res<-list(datos.centrados=datos.centrados,SIGMA.X=SIGMA.X,P=P,D=D,
Q=Q,lambdas=lambdas,scores=scores)
res
}
res<-compprinc1(x)

res

$datos.centrados
      [,1] [,2]
 [1,]    8    5
 [2,]    4    7
 [3,]    5    3
 [4,]    3   -1
 [5,]    2    5
 [6,]    1   -4
 [7,]    0    1
 [8,]   -1    3
 [9,]   -3   -6
[10,]   -5   -4
[11,]   -6   -6
[12,]   -8   -3

$SIGMA.X
         [,1]     [,2]
[1,] 23.09091 16.45455
[2,] 16.45455 21.09091

$P
```

```
             [,1]        [,2]
 [1,]  0.44916553 -0.23449074
 [2,]  0.37429410  0.30006853
 [3,]  0.27656981 -0.15814877
 [4,]  0.07278817 -0.35455124
 [5,]  0.23705083  0.28913890
 [6,] -0.09772429 -0.45821730
 [7,]  0.03326918  0.09273642
 [8,]  0.06445510  0.36548087
 [9,] -0.30567246 -0.29460372
[10,] -0.30983899  0.06541234
[11,] -0.41172982 -0.03278890
[12,] -0.38262716  0.41996358

$D
         [,1]     [,2]
[1,] 20.59937 0.000000
[2,]  0.00000 7.852774

$Q
          [,1]       [,2]
[1,] 0.7282381 -0.6853242
[2,] 0.6853242  0.7282381

$lambdas

[1] 38.575813  5.606005

$scores
            [,1]       [,2]
 [1,]  9.2525259 -1.8414027
 [2,]  7.7102217  2.3563703
 [3,]  5.6971632 -1.2419065
 [4,]  1.4993902 -2.7842106
 [5,]  4.8830971  2.2705423
 [6,] -2.0130586 -3.5982767
 [7,]  0.6853242  0.7282381
 [8,]  1.3277344  2.8700386
```

```
 [9,] -6.2966594 -2.3134563
[10,] -6.3824874  0.5136683
[11,] -8.4813738 -0.2574838
[12,] -7.8818776  3.2978790


#COMPONENTES PRINCIPALES CON BASE EN LA
#DESCOMPOSICION ESPECTRAL DE SIGMA:
compprinc2<-function(x,standar=0){
medias.x<-apply(x,2,mean)
medias<-t(medias.x)
uno<-c(rep(1,nrow(x)))
matriz.uno<-matrix(t(uno))
matriz.medias<-matriz.uno%*%medias
datos.centrados<-x-matriz.medias
SIGMA.X<-var(x)
R.X<-cor(x)
if(standar==1){
datos.estandarizados<-datos.centrados%*%
sqrt(solve(diag(diag(SIGMA.X))))
desc.espect<-eigen(R.X)
P<-desc.espect$vectors
lambdas<-desc.espect$values
lambdas1<-lambdas[lambdas>=1]
P1<-P[,1:length(lambdas1)]
scores<-datos.estandarizados%*%P1
R.LAMBDA<-sqrt(lambdas)
LOADINGS<-matrix(rep(0,(ncol(x)*ncol(x))),ncol=ncol(x))

for(i in 1:ncol(x)){
for(j in 1:ncol(x)){
LOADINGS[i,j]<-P[i,j]*R.LAMBDA[i]
}
}
res<-list(datos.estandarizados=datos.estandarizados,
SIGMA.X=SIGMA.X,R.X=R.X,lambdas=lambdas,P=P,P1=P1,
scores=scores,LOADINGS=LOADINGS)
res
```

```
}
else{ if(standar==0){
desc.espect<-eigen(SIGMA.X)
P<--desc.espect$vectors
lambdas<-desc.espect$values
scores<-datos.centrados%*%P
R.LAMBDA<-sqrt(lambdas)
STD<-sqrt(solve(diag(diag(SIGMA.X))))
AUX<-P%*%STD
LOADINGS<-matrix(rep(0,(ncol(x)*ncol(x))),ncol=ncol(x))

for(i in 1:ncol(x)){
for(j in 1:ncol(x)){
LOADINGS[i,j]<-AUX[i,j]*R.LAMBDA[i]
}
}
res<-list(datos.centrados=datos.centrados,SIGMA.X=SIGMA.X,
R.X=R.X,lambdas=lambdas,P=P,scores=scores,LOADINGS=LOADINGS)
res
}
}
}
res<-compprinc2(x)

res $datos.centrados
      [,1] [,2]
 [1,]    8    5
 [2,]    4    7
 [3,]    5    3
 [4,]    3   -1
 [5,]    2    5
 [6,]    1   -4
 [7,]    0    1
 [8,]   -1    3
 [9,]   -3   -6
[10,]   -5   -4
[11,]   -6   -6
[12,]   -8   -3
```

```
$SIGMA.X
         [,1]     [,2]
[1,] 23.09091 16.45455
[2,] 16.45455 21.09091

$lambdas

[1] 38.575813  5.606005

$P
          [,1]       [,2]
[1,] 0.7282381 -0.6853242
[2,] 0.6853242  0.7282381

$scores
            [,1]       [,2]
 [1,]  9.2525259 -1.8414027
 [2,]  7.7102217  2.3563703
 [3,]  5.6971632 -1.2419065
 [4,]  1.4993902 -2.7842106
 [5,]  4.8830971  2.2705423
 [6,] -2.0130586 -3.5982767
 [7,]  0.6853242  0.7282381
 [8,]  1.3277344  2.8700386
 [9,] -6.2966594 -2.3134563
[10,] -6.3824874  0.5136683
[11,] -8.4813738 -0.2574838
[12,] -7.8818776  3.2978790

$LOADINGS
          [,1]       [,2]
[1,] 0.9412618 -0.9268425
[2,] 0.3376776  0.3754503
```

utilizando los resultados obtenidos con $res < -compprinc2(x)$ se procede a graficar el "Scores Plot" para los datos de la matriz $\mathbf{X}$:

```
prin1<-res$scores[,1]
prin2<-res$scores[,2]

plot(prin1,prin2,main='SCORES COMPONENTES PRINCIPALES')
abline(h=0,lty=3)
abline(v=0,lty=3)
```

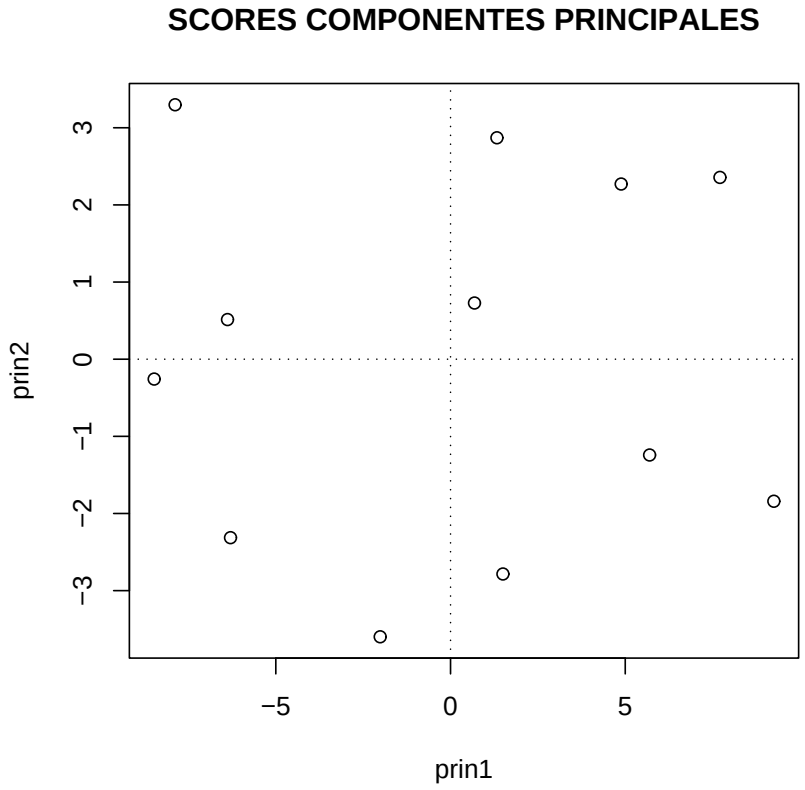

Figura 4.10:

Veamos otro Ejemplo: Los datos corresponden a los presentados en la tabla 4.7 p. 71 de Sharma S. (1996), sobre los precios en ctvs. de dólar por libra para cinco alimentos en 23 ciudades de E.U:

```
filas<-c("Atlanta","Baltimore","Boston","Buffalo","Chicago",
```

```
"Cincinnati","Cleveland","Dallas","Detroit","Honolulu",
"Houston","Kansas
C.","L.A", "Milwaukee","Minneapolis", "N.Y","Philadelphia",
"Pittsburg","St.Louis","San
Diego","San Fco","Seattle", "Washington D.C")

columnas<-c("Pan","Carne","Leche","Naranjas","Tomates")

pan<-c(24.5,26.5,29.7,22.8,26.7,25.3,22.8,23.3,24.1,29.3,
22.3,26.1,26.9,
20.3,24.6,30.8,24.5,26.2,26.5,25.5,26.3,22.5,24.2)
carne<-c(94.5,91,100.8,86.6,86.7,102.5,88.8,85.5,93.7,
105.9,83.6,88.9,
89.3,89.6,92.2,110.7,92.3,95.4,92.4,83.7,87.1,77.7,93.8)
leche<-c(73.9,67.5,61.4,65.3,62.7,63.3,52.4,62.5,51.5,
80.2,67.8,65.4,56.2,
53.8,51.9,66.0,66.7,60.2,60.8,57.0,58.3,62,66)
naranja<-c(80.1,74.6,104,118.4,105.9,99.3,110.9,117.9,
109.7,133.2,108.6,
100.9,82.7,111.8,106.0,107.3,98,117.1,115.1,92.8,
101.8,91.1,81.6)
tomate<-c(41.6,53.3,59.6,51.2,51.2,45.6,46.8,41.8,52.4,
61.7,42.4,43.2,
38.4,53.9,50.7,62.6,61.7,49.3,46.2,35.4,41.5,44.9,46.2)

precios<-matrix(c(pan,carne,leche,naranja,tomate),ncol=5,byrow=F)
precios
```

```
     [,1]  [,2] [,3]  [,4] [,5]
[1,] 24.5  94.5 73.9  80.1 41.6

[2,] 26.5  91.0 67.5  74.6 53.3

[3,] 29.7 100.8 61.4 104.0 59.6

[4,] 22.8  86.6 65.3 118.4 51.2

[5,] 26.7  86.7 62.7 105.9 51.2

[6,] 25.3 102.5 63.3  99.3 45.6

[7,] 22.8  88.8 52.4 110.9 46.8
```

```
 [8,] 23.3  85.5 62.5 117.9 41.8

 [9,] 24.1  93.7 51.5 109.7 52.4

[10,] 29.3 105.9 80.2 133.2 61.7

[11,] 22.3  83.6 67.8 108.6 42.4

[12,] 26.1  88.9 65.4 100.9 43.2

[13,] 26.9  89.3 56.2  82.7 38.4

[14,] 20.3  89.6 53.8 111.8 53.9

[15,] 24.6 92.2  51.9 106.0 50.7

[16,] 30.8 110.7 66.0 107.3 62.6

[17,] 24.5  92.3 66.7  98.0 61.7

[18,] 26.2  95.4 60.2 117.1 49.3

[19,] 26.5  92.4 60.8 115.1 46.2

[20,] 25.5  83.7 57.0  92.8 35.4

[21,] 26.3  87.1 58.3 101.8 41.5

[22,] 22.5  77.7 62.0  91.1 44.9

[23,] 24.2  93.8 66.0  81.6 46.2
```

```
res<-compprinc2(precios)#Procedimiento sin estandarizar datos
res $datos.centrados
```

```
             [,1]        [,2]        [,3]       [,4]        [,5]
 [1,] -0.791304348   2.6434783  11.6043478 -22.891304  -7.1652174

 [2,]  1.208695652  -0.8565217   5.2043478 -28.391304   4.5347826

 [3,]  4.408695652   8.9434783  -0.8956522   1.008696  10.8347826

 [4,] -2.491304348  -5.2565217   3.0043478  15.408696   2.4347826

 [5,]  1.408695652  -5.1565217   0.4043478   2.908696   2.4347826

 [6,]  0.008695652  10.6434783   1.0043478  -3.691304  -3.1652174

 [7,] -2.491304348  -3.0565217  -9.8956522   7.908696  -1.9652174

 [8,] -1.991304348  -6.3565217   0.2043478  14.908696  -6.9652174

 [9,] -1.191304348   1.8434783 -10.7956522   6.708696   3.6347826

[10,]  4.008695652  14.0434783  17.9043478  30.208696  12.9347826
```

```
[11,] -2.991304348  -8.2565217   5.5043478   5.608696  -6.3652174

[12,]  0.808695652  -2.9565217   3.1043478  -2.091304  -5.5652174

[13,]  1.608695652  -2.5565217  -6.0956522 -20.291304 -10.3652174

[14,] -4.991304348  -2.2565217  -8.4956522   8.808696   5.1347826

[15,] -0.691304348   0.3434783  -10.3956522  3.008696   1.9347826

[16,]  5.508695652  18.8434783   3.7043478   4.308696  13.8347826

[17,] -0.791304348   0.4434783   4.4043478  -4.991304  12.9347826

[18,]  0.908695652   3.5434783  -2.0956522  14.108696   0.5347826

[19,]  1.208695652   0.5434783  -1.4956522  12.108696  -2.5652174

[20,]  0.208695652  -8.1565217  -5.2956522 -10.191304 -13.3652174

[21,]  1.008695652  -4.7565217  -3.9956522  -1.191304  -7.2652174

[22,] -2.791304348 -14.1565217  -0.2956522 -11.891304  -3.8652174

[23,] -1.091304348   1.9434783   3.7043478 -21.391304  -2.5652174

$SIGMA.X
          [,1]     [,2]       [,3]        [,4]      [,5]
[1,]  6.284466 12.91097  5.7190514   1.3103755  7.285138

[2,] 12.910968 57.07711 17.5075296  22.6918775 36.294783

[3,]  5.719051 17.50753 48.3058893  -0.2750395 13.443478

[4,]  1.310375 22.69188 -0.2750395 202.7562846 38.762411

[5,]  7.285138 36.29478 13.4434783  38.7624111 57.800553

$R.X
           [,1]      [,2]         [,3]         [,4]      [,5]
[1,] 1.00000000 0.6817005  0.328238677  0.036709156 0.3822413

[2,] 0.68170049 1.0000000  0.333421671  0.210936756 0.6318983

[3,] 0.32823868 0.3334217  1.000000000 -0.002779124 0.2544167

[4,] 0.03670916 0.2109368 -0.002779124  1.000000000 0.3580615

[5,] 0.38224129 0.6318983  0.254416697  0.358061497 1.0000000

$lambdas

[1] 218.998679  91.723169  37.662690  20.810541   3.029229

$P
           [,1]       [,2]        [,3]        [,4]        [,5]
[1,] -0.02848905 -0.1653211 -0.02135748  0.18972574 -0.96716354
```

```
[2,] -0.20012240 -0.6321849 -0.25420475  0.65862454  0.24877074

[3,] -0.04167230 -0.4421503  0.88874949 -0.10765906  0.03606094

[4,] -0.93885906  0.3143547  0.12135003  0.06904699 -0.01521357

[5,] -0.27558389 -0.5279160 -0.36100184 -0.71684022 -0.03429221

$scores
             [,1]       [,2]        [,3]         [,4]       [,5]
 [1,]  22.4762714 -10.084571   9.4670699  3.897356034  2.4353740

 [2,]  25.3258177 -13.278372  -0.2650717 -6.106149325 -0.9179850

 [3,]  -5.8109806 -11.389537  -6.9526149 -0.873898110 -2.4582465

 [4,] -14.1398558   5.965021   5.0504380 -4.939610158  0.8922542

 [5,]  -2.4268891   2.477207   1.1140994 -4.716991085 -2.7583954

 [6,]   2.1657978  -6.663567  -1.1184854  8.917660329  2.8402938

 [7,]  -5.7885403  10.243124  -6.2953979  0.534410326  1.2393512

 [8,] -10.7573651  12.621018   6.1636282  1.435985628  0.3640076

 [9,]  -7.1853097   3.994880 -10.5358708  0.008045615  0.9947803

[10,] -35.5970593 -14.789443  11.2532912  0.896013891 -0.6409544

[11,]  -2.0034678   8.403846  10.0331906 -1.647960886  1.1705406

[12,]   3.9363863   2.643342   5.2485516  1.716954787 -1.1830315

[13,]  22.6269688   3.138735  -3.5224733  5.306826478 -1.7475252

[14,]  -8.7373757   6.066383  -7.6550168 -4.591150318  3.6495950

[15,]  -2.9737684   4.417981  -9.6450346  0.035062561  0.2670537

[16,] -11.9402096 -20.410290  -6.0870379  3.437285891 -1.0464971

[17,]   0.8717678 -10.494444  -1.4566474 -9.949019978  0.6668433

[18,] -14.0411408   2.689048  -1.2636531  3.322649890 -0.3058972

[19,] -10.7422955   5.278547   0.9022092  3.423209238 -1.1839890

[20,] 15.0984763   11.315431   0.9506142  4.114680957 -1.8085441

[21,]  4.2103016    8.067854   0.1146454  2.614525770 -2.0356797

[22,] 15.1543297    7.844146   3.3478500 -7.871900765 -0.5192861

[23,] 20.2781405   -8.056339   1.1517162  1.036013231  2.0859376
```

```
$LOADINGS
           [,1]       [,2]        [,3]        [,4]         [,5]
[1,] -0.1681762 -0.3238307 -0.04547479  0.19717863 -1.882585314

[2,] -0.7645414 -0.8014061 -0.35028629  0.44298628  0.313381123

[3,] -0.1020162 -0.3591655  0.78475662 -0.04640012  0.029108965

[4,] -1.7084721  0.1898151  0.07964922  0.02212072 -0.009128654

[5,] -0.1913312 -0.1216186 -0.09040146 -0.08761960 -0.007850469
```

```
res<-compprinc2(precios,1)#Procedimiento estandarizando datos
res $datos.estandarizados
```

```
             [,1]        [,2]        [,3]        [,4]        [,5]
 [1,] -0.31565258  0.34990064  1.66963176 -1.60761991 -0.94246097

 [2,]  0.48215065 -0.11337241  0.74880075 -1.99387617  0.59647257

 [3,]  1.75863582  1.18379214 -0.12886630  0.07083909  1.42512909

 [4,] -0.99378533 -0.69577282  0.43226510  1.08212820  0.32025373

 [5,]  0.56193097 -0.68253645  0.05817750  0.20427307  0.32025373

 [6,]  0.00346871  1.40881048  0.14450541 -0.25923444 -0.41632985

 [7,] -0.99378533 -0.40457262 -1.42378489  0.55541512 -0.25849051

 [8,] -0.79433452 -0.84137292  0.02940153  1.04701400 -0.91615441

 [9,] -0.47521323  0.24400966 -1.55327675  0.47114103  0.47809306

[10,]  1.59907517  1.85884716  2.57607478  2.12150867  1.70134793

[11,] -1.19323614 -1.09286401  0.79196471  0.39388978 -0.83723474

[12,]  0.32259000 -0.39133624  0.44665308 -0.14686898 -0.73200852

[13,]  0.64171130 -0.33839075 -0.87704149 -1.42502605 -1.36336587

[14,] -1.99103937 -0.29868163 -1.22235311  0.61862069  0.67539224

[15,] -0.27576242  0.04546406 -1.49572482  0.21129591  0.25448734

[16,]  2.19742760  2.49419306  0.53298099  0.30259285  1.81972743

[17,] -0.31565258  0.05870044  0.63369688 -0.35053137  1.70134793

[18,]  0.36248016  0.46902800 -0.30152211  0.99083127  0.07034144

[19,]  0.48215065  0.07193681 -0.21519420  0.85037445 -0.33741018

[20,]  0.08324903 -1.07962764 -0.76193761 -0.71571910 -1.75796421
```

```
[21,]  0.40237033 -0.62959096 -0.57489381 -0.08366341 -0.95561425

[22,] -1.11345581 -1.87381001 -0.04253839 -0.83510740 -0.50840279

[23,] -0.43532307  0.25724603  0.53298099 -1.50227730 -0.33741018

$SIGMA.X
          [,1]     [,2]       [,3]        [,4]      [,5]
[1,]  6.284466 12.91097  5.7190514   1.3103755  7.285138

[2,] 12.910968 57.07711 17.5075296  22.6918775 36.294783

[3,]  5.719051 17.50753 48.3058893  -0.2750395 13.443478

[4,]  1.310375 22.69188 -0.2750395 202.7562846 38.762411

[5,]  7.285138 36.29478 13.4434783  38.7624111 57.800553

$R.X
           [,1]      [,2]         [,3]         [,4]      [,5]
[1,] 1.00000000 0.6817005  0.328238677  0.036709156  0.3822413

[2,] 0.68170049 1.0000000  0.333421671   0.210936756 0.6318983

[3,] 0.32823868 0.3334217  1.000000000  -0.002779124 0.2544167

[4,] 0.03670916 0.2109368 -0.002779124  1.000000000  0.3580615

[5,] 0.38224129 0.6318983  0.254416697  0.358061497  1.0000000

$lambdas

[1] 2.4224680 1.1046749 0.7384805 0.4936113 0.2407653

$P
          [,1]        [,2]        [,3]        [,4]         [,5]
[1,] 0.4961487 -0.30861972  0.38639398  0.50930459 -0.499898868

[2,] 0.5757023 -0.04380176  0.26247227 -0.02813712  0.772635014

[3,] 0.3395696 -0.43080905 -0.83463952  0.04910000  0.007882237

[4,] 0.2249898  0.79677694 -0.29160659  0.47901574 -0.005966796

[5,] 0.5064340  0.28702846  0.01226602 -0.71270629 -0.391201387

$P1
          [,1]        [,2]
[1,] 0.4961487 -0.30861972

[2,] 0.5757023 -0.04380176

[3,] 0.3395696 -0.43080905

[4,] 0.2249898  0.79677694

[5,] 0.5064340  0.28702846
```

```
$scores
            [,1]        [,2]
 [1,] -0.2272083 -2.18862973

 [2,]  0.2816918 -1.88389539

 [3,]  2.2479696 -0.07358954

 [4,] -0.3411840  1.10509084

 [5,]  0.1137656  0.08609193

 [6,]  0.5926774 -0.45108366

 [7,] -1.2153983  1.30614999

 [8,] -1.0969088  0.84060868

 [9,] -0.2746205  1.31775844

[10,]  4.0772166  0.49398124

[11,] -1.2876427  0.14847209

[12,] -0.3173288 -0.60196771

[13,] -1.1853159 -1.33213794

[14,] -1.0936522  1.84091707

[15,] -0.4421275  0.96888693

[16,]  3.6967998 -0.25361999

[17,]  0.8751220  0.03088303

[18,]  0.6060271  0.80714718

[19,]  0.2280088  0.52146789

[20,] -1.8902935 -0.72500732

[21,] -0.8608169 -0.18988233

[22,] -2.0910043 -0.36728388

[23,] -0.3957768 -1.40035782

$LOADINGS
          [,1]        [,2]        [,3]        [,4]         [,5]
[1,] 0.7722197 -0.48034435  0.60139440  0.79269590 -0.778056569

[2,] 0.6050834 -0.04603719  0.27586758 -0.02957311  0.812066560

[3,] 0.2918087 -0.37021529 -0.71724656  0.04219403  0.006773591
```

```
[4,] 0.1580722  0.55979539 -0.20487544  0.33654437 -0.004192120

[5,] 0.2484962  0.14083867  0.00601867 -0.34970959 -0.191954075
```

Para los resultados obtenidos estandarizando los datos se construyen a continuación el "Scree Plot", el "Scores Plot", y el "Principal Components Pattern Plot", como sigue:

```
plot(res$lambdas,type='o',pch=19,xlab="Nro. de Componentes
Principales",ylab="eigenvalores",main="SCREE PLOT\nDATOS
PRECIOS DE\nCINCO ALIMENTOS EN
23 CIUDADES DE E.U",cex.main=0.8)

prin1<-res$scores[,1] prin2<-res$scores[,2]

plot(prin1,prin2,pch=19,cex=0.5,ylim=c(min(prin2)-0.5,
max(prin2)+0.5),xlim=c(min(prin1)-0.5,max(prin1)+1),
main="SCORES
COMPONENTES PRINCIPALES DATOS PRECIOS DE\nCINCO
ALIMENTOS EN 23 CIUDADES DE
E.U",cex.main=0.8) text(prin1,prin2,labels=filas,
adj=c(0,1),cex=0.7)
abline(h=0,lty=3)
abline(v=0,lty=3)

plot(res$LOADINGS[,1],res$LOADINGS[,2],pch=19,cex=0.5,
xlim=c(-1,1),ylim=c(min(res$LOADINGS[,2]),
max(res$LOADINGS[,2])),main='PATRON PARA LAS
DOS PRIMERAS COMPONENTES\nDATOS ESTANDARIZADOS\nPRECIOS
DE CINCO
ALIMENTOS',cex.main=0.8,xlab='prin1',ylab='prin2',cex.axis=0.8)
text(res$LOADINGS[,1],res$LOADINGS[,2],labels=columnas,
adj=c(0,1),cex=0.7)
abline(h=0,lty=3)
abline(v=0,lty=3)
```

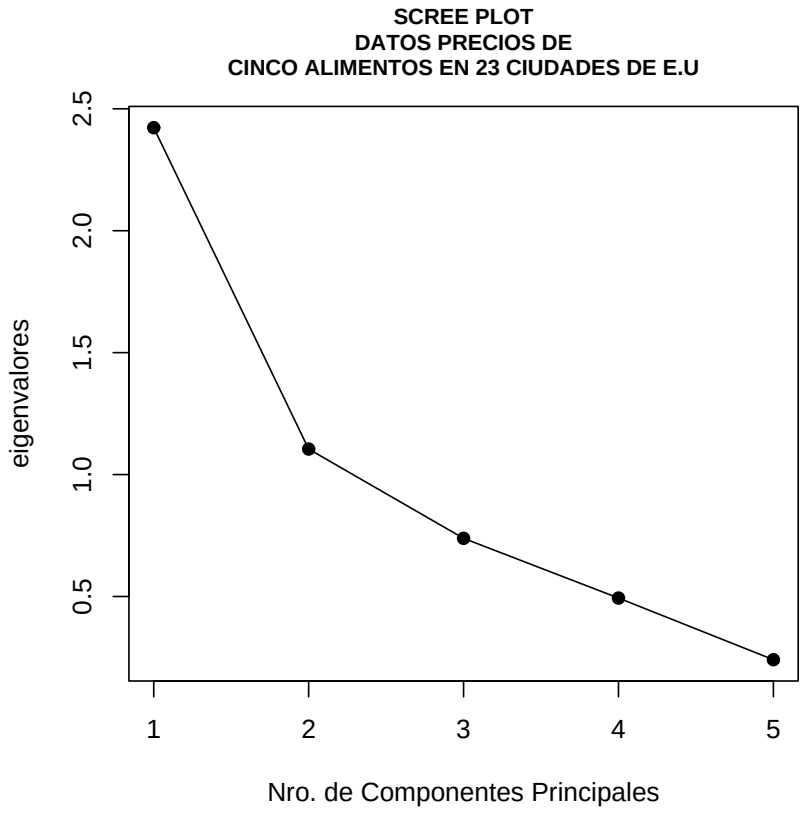

Figura 4.11: *El codo en la gráfica indica que el número de componentes a conservar debe ser dos. La primera componente (analizar la columna 1 de la matriz P), puede intepretarse como el grupo de "no frutas" y la segunda componente como "frutas"*

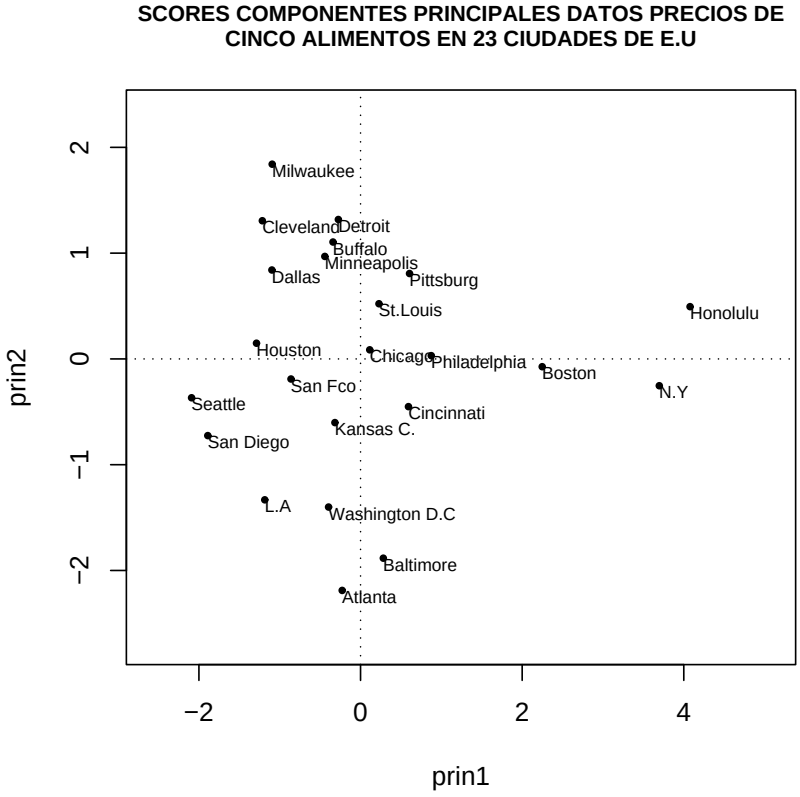

Figura 4.12: *cinco posibles grupos de ciudades pueden ser identificados e interpretarse dichos agrupamientos en función del costo de las frutas y no frutas en dichas ciudades en función de la distancia a los ejes que pasan por el punto* $(0, 0)$ *(puntos cercanos al eje vertical implican que en esas ciudades las "no frutas" tienen un bajo costo y puntos cercanos al eje horizontal indican que las "frutas" tienen un precio bajo).*

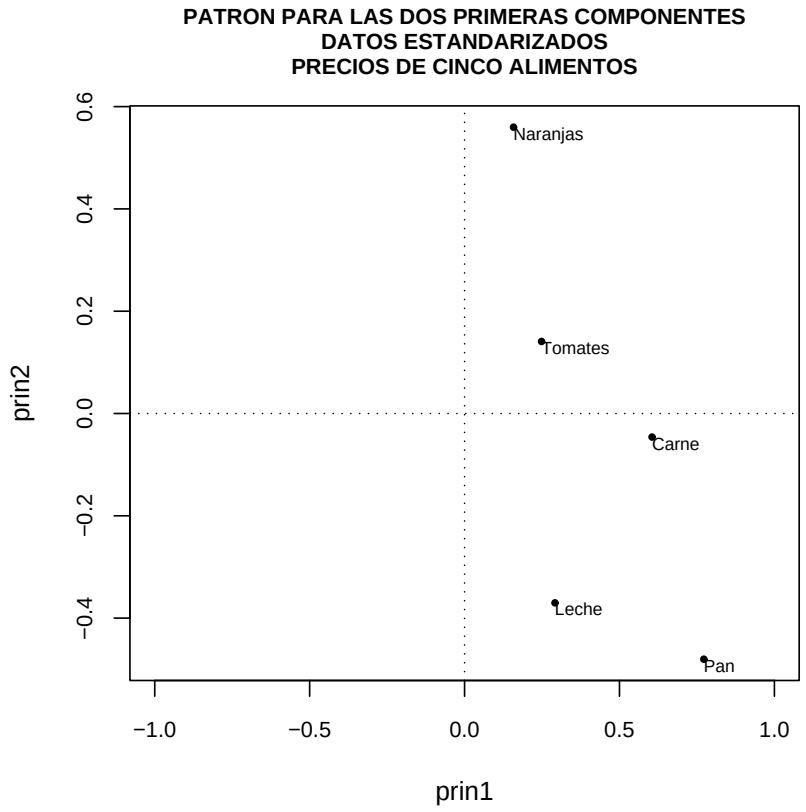

Figura 4.13: *De acuerdo con la cercanía de los puntos a cada eje es posible identificar gráficamente cuáles variables tienen mayor peso en cada componente, y por tanto, según la naturaleza de éstas, tratar de interpretar lo que las componentes resultantes pueden representar. Para el caso, la primera componente parece estar conformada por los alimentos que no son frutas y la segunda por las frutas.*

4.4 Anális de Agrupamientos (Clusters)

Para realizar análisis cluster en R es necesario cargar los siguientes dos paquetes:

```
library(mva)

library(cluster)
```

Aunque también en el paquete survival existe una función 'cluster'. A continuación se listan las funciones relacionadas con el análisis cluster disponibles en R; el nombre entre paréntesis corresponde al del paquete desde el cual debe invocarse:

- **clara.object(cluster)**: Crea un objeto "Clustering Large Applications (CLARA) Object" que representa el particionamiento de una gran base de datos en clusters.
- **clara(cluster)**: Calcula un objeto clase "clara" (Clustering Large Applications), es decir una lista que representa un "clustering" de los datos en 'k' clusters.
- **clusplot.default(cluster)**: Gráfico de cluster bivariado (clusplot)
- **clusplot(cluster)**: Función genérica que realiza un Cluster "clusplot" de dos dimensiones sobre el dispositivo gráfico actual. Posee un método por defecto y 'partition'.
- **plot.agnes(cluster)**: Gráficos de un clustering jerárquico aglomerativo. es decir resultante de 'agnes'.
- **plot.diana(cluster)**: Gráficos de un clustering divisivo, es decir resultante de 'diana'
- **plot.mona(cluster)**: *Banner* de clusterings *Monothetic Divisive Hierarchical*
- **pltree(cluster)**: Función genérica que dibuja "Clustering Trees"

- `pltree.twins(cluster)`: "Clustering Tree" de un clustering jerárquico, típicamente resultante de 'agnes' o 'diana'.
- **twins.object(cluster)**: Objeto clustering jerárquico.
- **xclara(cluster)**: Conjunto de datos bivariados con 3 clusters.
- **hclust(mva)**: Clustering jerárquico.
- **identify.hclust(mva)**: Identifica clusters en un dendrograma.
- **kmeans(mva)**: Clustering de una matriz por el algoritmo 'K-Means'.
- **rect.hclust(mva)**: Dibuja rectángulos alrededor de clusters jerárquicos.
- **cluster(survival)**: Identifica Clusters.

4.4.1 Funciones `clusplot, clusplot.default y clusplot.partition`

`clusplot` es una función genérica que dibuja un "clusplot" de dos dimensiones sobre el dispositivo gráfico actual. Posee un método por defecto ('clusplot.default) y 'partition' ('clusplot.partition').

```
clusplot(x, ...)
```

Argumentos:

- x: Un objeto R.
- ...: Argumentos adicionales para 'methods'. También pueden proporcionarse parámetros gráficos (ver 'par') como argumentos a esta función.

`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 multidimensional. Una elipse es dibujada alrededor de cada cluster.

```
clusplot.default(x, clus, diss = FALSE, cor = TRUE,
stand = FALSE,lines = 2, shade =
FALSE, color = FALSE, labels= 0, plotchar = TRUE,
col.p = "dark green", col.txt = col.p,
span = TRUE, xlim = NULL, ylim = NULL, ...)
```

Argumentos:

- x: matriz o marco de datos, o matriz de disimililaridad, 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. Valores faltantes ('NA') son permitidos, los cuales son reemplazados por la mediana de la variable correspondiente. Cuando algunas variables o algunas observaciones contienen sólo valores faltantes, la función para con un mensaje de advertencia.

  En caso de ser un matriz de disimilaridad, 'x' es el resultado de la función 'daisy' o 'dist' o es una matriz simétrica. También 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.

- clus: Un vector de longitud n que representa un clustering de 'x'. Para cada obseración 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'.

- diss: Argumento lógico que indica si 'x' será considerada como una matriz de disimilaridad o una matriz de observaciones por variables.

- cor: Argumento lógico, usado sólo cuando se trabaja con una matriz de datos ('diss = FALSE'). Si es TRUE las variables son escaladas a una varianza unitaria.

- stand: Argumento lógico; si es TRUE las representaciones de las n observaciones en el gráfico de dos dimensiones son estandarizadas.

- lines: 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 linea que conecta sus centros. Si las elipses se sobreponen en la linea 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 linea de distancia.
  - lines = 1, aparecerá el segmento de linea que une los centros de las elipses.
  - lines = 2, es dibujado un segmento de linea entre los bordes de las elipses.

- shade: Argumento lógico; si es TRUE, 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.

- color: Argumento lógico; si es TRUE, las elipses son coloreadas con respecto a sus densidades. A medida que incrementa la densidad, los colores son azul claro, verde claro, rojo y púrpura. Cuando se tienen 4 o menos clusters, entonces '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.

- labels: Código de valor entero, actualmente sólo uno de 0,1,2,3 y 4:

  - labels= 0,no se colocan etiquetas en el gráfico.
  - labels= 1, puntos y elipses pueden ser identificados en el gráfico (ver 'identify').
  - labels= 2, Todos los puntos y elipses son identificados en el gráfico.
  - labels= 3, sĺo los puntos son etiquetados.
  - labels= 4, sĺo las elipses son etiquetadas.

  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, ('diss = TRUE'), 'x'es un vector, y las etiquetas de los puntos pueden ser vinculados a 'x' como

un atributo “labels” (‘attr(x,”Labels”)’), como sucede para la salida de la función ‘daisy’. No deberá tomarse en cuenta un posible atributo ‘names’ de ‘clus’.

- plotchar: Argumento lógico; si es TRUE los símbolos de graficación serán diferentes para los puntos que pertenecen a clusters diferentes.

- span: Argumento lógico; si es TRUE cada cluster es representado por la elipse con el áreas más pequeña que contenga a todos sus puntos (esta es un caso especial del elipsoide de volumen mínimo. Si es FALSE la elipse está basada en la media y la matriz de covarianza de sus puntos, a menudo produce una elipse mucho mayor.

  también hay 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 linea recta, con ‘span=FALSE’ dibuja una elipse angosta alrededor de ésta y si ‘span=TRUE’ dibuja el segmento de linea exacto.

- col.p: Código de color usado para los puntos de las observaciones. color code used for the observation points.

- col.txt: Código de color usado para las etiquetas.

- xlim, ylim: Los rangos de x e y como en ‘plot.default’.

- ...: Pueden especificarse parámetros gráficos adicionales, ver ‘par’.

Detalles: ‘clusplot’ usa las funciones ‘princomp’ y ‘cmdscale’, que corresponden a técnicas de reducción de datos y que representarán a los datos en un gráfico bivariado.

Valor: Una lista invisible con los siguientes componentes:

- Distances: 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.

- Shading: 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 linea, $y[i]$ y la densidad del clusters son ajustados a NA. Sea z la suma de todos los elementos de y sin los NA's, entonces el sombreado es igual a $y/z * 37 + 3$

`clusplot.partition` realiza un gráfico bivariado clusplot (Clustering Plot) de un objeto tipo partición.

```
clusplot.partition(x, ...)
```

Argumentos:

- x: Un objeto de clase "partition", es decir, creado con las funciones 'pam', 'clara', o 'fanny'.
- ...: Todos los argumentos opcionales disponibles para 'clusplot.default' (excepto los correspondientes a 'diss') pueden proporcionarse para esta función. También pueden proporcionarse parámetros gráficos (ver '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 dad 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 o está disponible un matriz de disimilaridad.

Valor: Una lista invisible como en 'clusplot.default'.
Referencias

- Kaufman, L. and Rousseeuw, P.J. (1990). Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York.

- Pison, G., Struyf, A. and Rousseeuw, P.J. (1997). Displaying a Clustering with CLUSPLOT, Technical Report, University of Antwerp, submitted.

- Struyf, A., Hubert, M. and Rousseeuw, P.J. (1997). Integrating Robust Clustering Techniques in S-PLUS, Computational Statistics and Data Analysis, 26, 17-37.

Ejemplo 1: El siguiente ejemplo está basado en las observaciones disponibles en data(votes.repub) sobre el porcentaje de votos por el partido republicano en cada estado en E.U desde 1856 a 1976.

```
library(mva)
library(cluster)
data(votes.repub)

#Calcular disimilaridades
#(con base en la distancia euclidiana):

votes.diss <- daisy(votes.repub)

#Particionar en dos clusters los estados
#usando la matriz de disimilaridades y solicitar
#la componente clustering.
#'pam' minimiza la suma de las
#disimilaridades:

votes.clus <- pam(votes.diss, 2, diss = TRUE)$clustering

#Graficar los clusters identificando
#interactivamente puntos deseados

#haciendo click con el mouse sobre estos;
#labels debe ser igual a 1:

if(interactive())
clusplot(votes.diss, votes.clus, lines=1,diss = TRUE,
```

```
color=TRUE,col.p="black",labels=1)
```

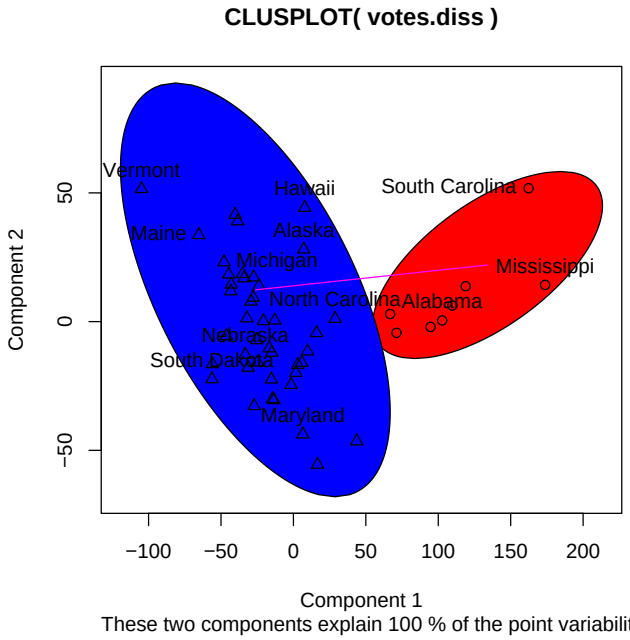

Figura 4.14: *Partición en dos clusters de los estados de E.U, según los datos provistos en data(votes.repub) sobre el porcentaje de votos en cada estado por el partido republicano desde 1856 hasta 1976, utilizando disimilaridades con base en la distancia euclidiana. Inspeccionando la base de datos y comparando con las observaciones en cada cluster, puede deducirse que el cluster de la izquierda está constituído por estados donde hay mayor preferencia por el partido republicano y el de la izquierda básicamente estados del sur en donde hay menor acogida por dicho partido.*

Ejemplo 2: Generar 45 objetos y dividir en dos clusters:

```
x <- rbind(cbind(rnorm(20,0,0.5), rnorm(20,0,0.5)),
cbind(rnorm(25,5,0.5),rnorm(25,5,0.5)))
```

```
clusplot(pam(x, 2),lines=1,color=TRUE,col.p="black",labels=4)
```

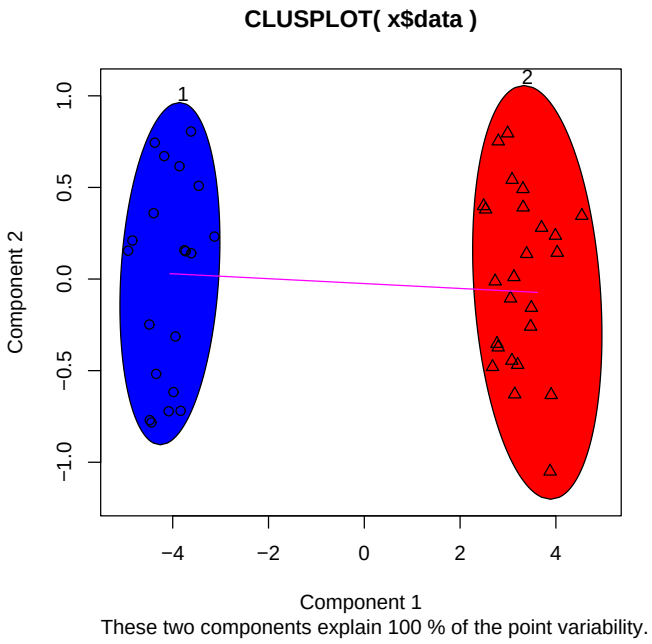

Figura 4.15: *Partición en dos clusters de datos simulados en donde 20 provienen de una normal $\mu = 0$, $\sigma = 0.5$ (agrupadas en la elipse 1) y 25 provienen de una normal $\mu = 5$, $\sigma = 0.5$ (agrupadas en la elipse 2). Observe la distancia entre las elipses lo que indica una clara diferencia entre las observaciones en el cluster 1 y las pertenecientes al cluster 2.*

4.4.2 Función `hclust`

Esta función desarrolla un clustering jerárquico sobre un conjunto de disimilaridades.

```
hclust(d, method = "complete", members=NULL)
plot.hclust(tree, labels = NULL, hang = 0.1,
```

```
axes = TRUE, frame.plot = FALSE, ann = TRUE,
main = "Cluster Dendrogram", sub = NULL,
xlab = NULL, ylab = "Height", ...)
```

Argumentos:

- d: Una estructura de disimilaridad como la producida por 'dist'.
- method: El método de aglomeración a ser usado. Puede ser uno de los siguientes 8o una abeviación no ambigua): "ward", "single", "complete", "average", "mcquitty", "median" o "centroid".
- members: 'NULL' o un vector de longitud 'd'.
- tree: Un objeto del tipo producido por 'hclust'.
- hang: La fracción de la altura de gráfico por la cual las etiquetas deberán pender abajo del resto del gráfico. Un valor negativo provocará que las etiquetas pendan hacia abajo de cero.
- labels: Un vector de caracteres de etiquetas para las hojas del árbol. Por defecto, los nombres de las filas o los números de las filas de los datos originales son usados. SI 'labels=FALSE' ninguna etiqueta aparecerá en el gráfico.
- axes, frame.plot, ann: Argumentos lcomo en 'plot.default'.
- main, sub, xlab, ylab: Cadena de caracteres para 'title'. 'sub' y 'xlab', tienen por defecto un valor no NULL cuando hay un 'tree$call'.
- ...: Pueden especificarse argumentos gráficos adicionales.

Detalles: Esta función desarrolla un análisis cluster usando un conjunto de disimilaridades para los 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 clusters. En cada etapa las distancias entre clusters son recalculadas por la fórmula actual de disimilaridad de Lance-Williams según el método de clustering que sea usado.

Diferentes métodos de cluster son proporcionados: 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 otros métodos pueden ser considerados como apuntando al hallazgo de clusters con caracter´ticas entre los métodos de linkeo 'single' y 'complete'.

Si 'members!=NULL', entonces 'd' es tomado como una matriz de disimilaridad entre clusters en vez de disimilaridades entre *singletons* y 'members' da el número de observaciones por cluster. De esta forma el algoritmo de cluster jerárquico puede ser iniciado en el medio del dendrograma, es decir, con el fin de reconstruir la parte del árbol arriba de un corte. Las disimilaridades entre clusters pueden ser calculadas eficientemente (es decir, sin 'hclust' en sí) sólo por un número limitado de combinaciones de distancia/linkeo, lo más simple entre el cuadrado de la distancia euclidiana y linkeo de centroides. En este caso las disimilaridades entre clusters son los cuadrados de las distancias euclidianas entre las medias de clusters.

En los despliegues de clusters jerárquicos, es necesaria una decisión en cada fusión para especificar cuál sub árbol deberá ir a la izquierda y cuál a la derecha. Dado que para n observaciones hay n-1 fusiones, hay $2^{(n-1)}$ posibles ordenamientos para las hojas en un árbol cluster o dendrograma. El algoritmo usado en 'hclust' consiste en ordenar el su árbol de manera que el cluster más estrecho quede a la izquierda (el último, o sea el más reciente, la fusión del sub árbol izquierdo está en un valor inferior al de la última fusión del subárbol de la derecha). Las Observaciones individuales son los cluster más estrechos posibles, y las fusiones incluyen dos observaciones colocadas en orden según sus números de secuencia de observación.

Valor: Esta función produce un objeto de clase hclust que describe el árbol producido por el proceso clustering. El objeto es una lista con los siguientes componentes:

- merge: Es una matriz de n-1 por 2. La fila i describe la fusión de los cluster en el paso i del proceso. Si un elemento j en la fila es negativo, significa que la observación -j fue fusionada en esta etapa. Si j es positivo entonces la fución fué con el cluster formado en la etapa (anterior) j del algoritmo. Así las entradas negativas en 'merge' indican aglomeraciones de *singletons*, y entradas positivas indican aglomeraciones de *non-singletons*.

- height: Un conjunto de n-1 valores reales decrecientes. Corresponde al valor del criterio asociado con el método ('method') clustering para la aglomeración particular.

- order: Un vector que da la permutación de las observaciones originales apropiada para graficación, en el sentido que un gráfico cluster que use este ordenamiento y la matriz 'merge' no tendrá ramas cruzadas.

- labels: Etiquetas para cada uno de los objetos que están siendo agrupados.

- call: El call el cual produjo el resultado.

- method: El método cluster que ha sido usado.

- dist.method: La distancia que ha sido usada para crear a 'd' (sólo es devuelta si el objeto distancia tiene un atributo ' "method" ').

Autores: The 'hclust' function is based on Fortran code contributed to STATLIB by F. Murtagh.

Referencias:

- Everitt, B. (1974). Cluster Analysis. London: Heinemann Educ. Books.

- Hartigan, J. A. (1975). Clustering Algorithms. New York: Wiley.

- Sneath, P. H. A. and R. R. Sokal (1973). Numerical Taxonomy. San Francisco: Freeman.

- Anderberg, M. R. (1973). Cluster Analysis for Applications. Academic Press: New York.

- Gordon, A. D. (1981). Classification. London: Chapman and Hall.
- Murtagh, F. (1985). "Multidimensional Clustering Algorithms", in COMPSTAT Lectures 4. Wuerzburg: Physica-Verlag (for algorithmic details of algorithms used).

ejemplo: Hacer clusters por el método "average" utilizando como disimilaridades las distancias euclidianas y graficar el dendrograma o árbol:

```
library(mva)
data(USArrests)
```

```
USArrests
               Murder Assault UrbanPop Rape
Alabama          13.2     236       58 21.2
Alaska           10.0     263       48 44.5
Arizona           8.1     294       80 31.0
Arkansas          8.8     190       50 19.5
California        9.0     276       91 40.6
Colorado          7.9     204       78 38.7
Connecticut       3.3     110       77 11.1
Delaware          5.9     238       72 15.8
Florida          15.4     335       80 31.9
Georgia          17.4     211       60 25.8
Hawaii            5.3      46       83 20.2
Idaho             2.6     120       54 14.2
Illinois         10.4     249       83 24.0
Indiana           7.2     113       65 21.0
Iowa              2.2      56       57 11.3
Kansas            6.0     115       66 18.0
Kentucky          9.7     109       52 16.3
Louisiana        15.4     249       66 22.2
Maine             2.1      83       51  7.8
Maryland         11.3     300       67 27.8
Massachusetts     4.4     149       85 16.3
Michigan         12.1     255       74 35.1
Minnesota         2.7      72       66 14.9
Mississippi      16.1     259       44 17.1
Missouri          9.0     178       70 28.2
Montana           6.0     109       53 16.4
Nebraska          4.3     102       62 16.5
Nevada           12.2     252       81 46.0
New Hampshire     2.1      57       56  9.5
New Jersey        7.4     159       89 18.8
New Mexico       11.4     285       70 32.1
New York         11.1     254       86 26.1
North Carolina   13.0     337       45 16.1
North Dakota      0.8      45       44  7.3
Ohio              7.3     120       75 21.4
Oklahoma          6.6     151       68 20.0
```

```
Oregon           4.9     159       67 29.3
Pennsylvania     6.3     106       72 14.9
Rhode Island     3.4     174       87  8.3
South Carolina  14.4     279       48 22.5
South Dakota     3.8      86       45 12.8
Tennessee       13.2     188       59 26.9
Texas           12.7     201       80 25.5
Utah             3.2     120       80 22.9
Vermont          2.2      48       32 11.2
Virginia         8.5     156       63 20.7
Washington       4.0     145       73 26.2
West Virginia    5.7      81       39  9.3
Wisconsin        2.6      53       66 10.8
Wyoming          6.8     161       60 15.6
```

```
#crear un objeto clase hclust

hc <- hclust(dist(USArrests), "ave")

#Graficar el dendrograma:

plot(hc,cex=0.6, main="Dendrograma de Clusters\ndata(USArrests)",
frame.plot=TRUE,
hang=-1)
```

Ejemplo 2: Hacer clusters con por el método "centroid" utilizando como disimilaridad el cuadrado de la distancia euclidiana, cortar el árbol en 10 clusters y reconstruir la parte superior del árbol a partir de los centros de clusters:

```
hc <- hclust(dist(USArrests)^2, "cen")
memb <- cutree(hc, k=10)
cent <- NULL

for(k in 1:10){
cent <- rbind(cent,apply(USArrests[memb==k,,drop=FALSE], 2, mean))
}

hc1 <- hclust(dist(cent)^2, method="cen", members=table(memb))

opar <- par(mfrow=c(1,2))
```

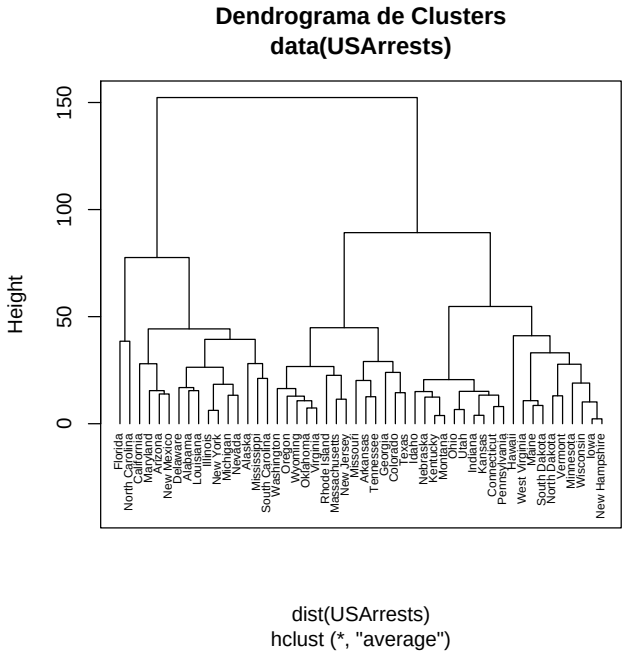

Figura 4.16: *Los datos en data(USArrests) han sido agrupadas mediante la función 'hclust' utilizando el método "average" sobre la matriz de distancias euclidianas. Pueden vislumbrarse 3 clusters principales.*

```
plot(hc,  labels=FALSE, hang=0.5, main= "Árbol
original",cex.main=0.8,cex.axis=0.7,cex.lab=0.7,frame.plot = TRUE)

plot(hc1, cex=0.7, hang=0.5,main= "Reiniciado desde cluster
10",cex.main=0.8,cex.axis=0.7,cex.lab=0.7,frame.plot = TRUE)

par(opar)
```

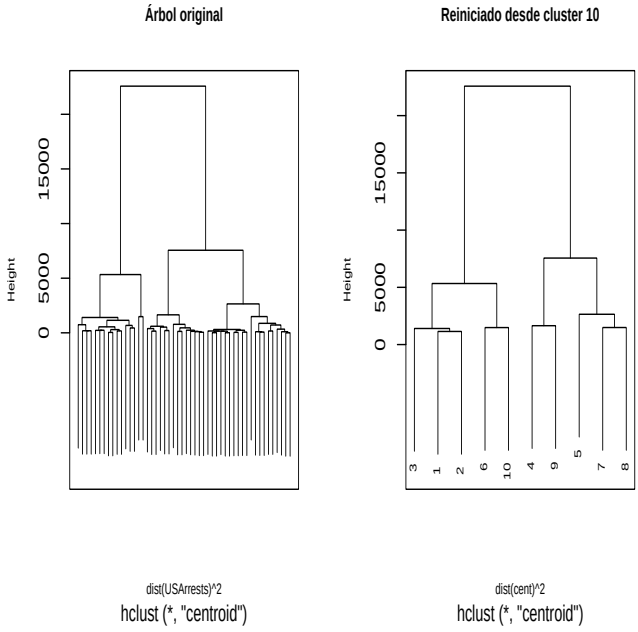

Figura 4.17: *La función 'cutree' corta un árbol o dendrograma en un número especificado de grupos, combinado con 'hclust' (aplicado este último sobre los centros de clusters) es posible extraer una parte particular de un gran árbol para verlo en detalle.*

4.4.3 Función `rect.hclust`

Esta función, disponible en el paquete 'mva', dibuja rectaángulos alrededor de las ramas de un dendrograma de clusters jerárquicos, resaltando los correspondientes clusters. Primero el dendrograma es cortado a cierto nivel, luego un rectángulo es dibujado alrededor de las ramas seleccionadas.

```
rect.hclust(tree, k = NULL, which = NULL, x = NULL,
h = NULL, border = 2, cluster = NULL)
```

Argumentos:

- tree: Un objeto del tipo producido por 'hclust'.
- k, h: Escalar. Corta el dendrograma de modo que exactamente 'k' clusters son producidos o corta a la altura 'h'.
- which, x: Un vector que selecciona los clusters alrededor de los cuales un rectángulo será dibujado. 'which' selecciona clusters por el número (de izquierda a derecha en el árbol), 'x' selecciona los clusters conteniendo las respectivas coordenadas horizontales. El valor por defecto es 'which =1:k'.
- border: Vector para especificar los colores de los bordes de los rectángulos.
- cluster: Un vector opcional con los miembros de cluster como el retornado por 'cutree(hclust.obj, k = k)', puede ser especificado.

Valor: (Invisible) retorna una lista donde cada elemento contiene un vector de los puntos de los datos contenidos en el cluster respectivo.

Ver: 'identify.hclust'.

Ejemplo 1: Usando los datos disponibles en data(USArrests) en el paquete 'mva', realizar un clustering jerárquico y resaltar en el árbol resultante los tres clusters principales:

```
library(mva)
data(USArrests)

#calculando clusters con m'etodo "complete"
#sobre matriz de distancias euclidianas:

hca <- hclust(dist(USArrests))

#Graficando el dendrograma:

plot(hca,cex=0.6,main="Dendrograma de Clusters\ndata
(USArrests)",hang=-1)

#Dibujando rectángulos de modo que
#el dendrograma quede cortado

#en exactamente 3 clusters:

rect.hclust(hca, k=3, border="black")
```

Ejemplo 2: Con los datos anteriores realizar un clustering jerárquico y resaltar en el árbol resultante en los clusters 2 y 7 (de izquierda a derecha):

```
#Graficar el dendrograma y
#cortar a la altura 50 los
#clusters 2 y 7:

plot(hca,cex=0.6,main="Dendrograma de Clusters\ndata
(USArrests)",hang=-1)

x <- rect.hclust(hca, h=50, which=c(2,7), border="black") x
```

```
 [[1]]
      Alabama        Alaska      Delaware     Louisiana    Mississippi
            1             2             8            18            24
South Carolina
           40
```

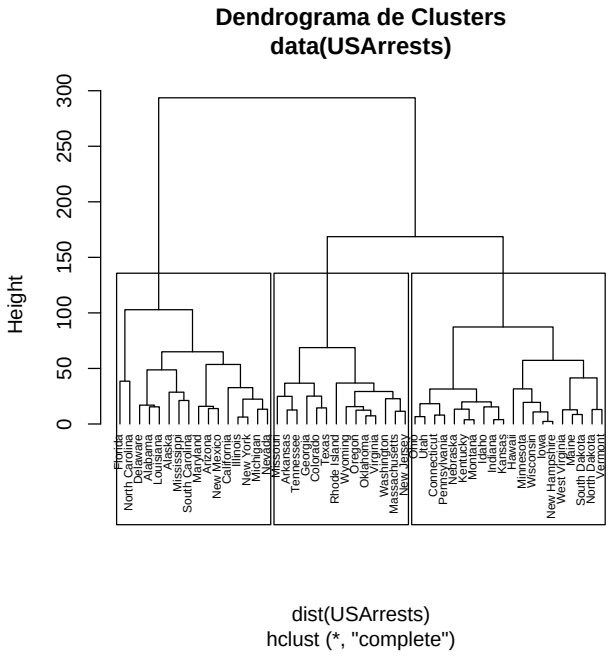

Figura 4.18: *La función 'rect.hclust' resalta en la gráfica los 3 clusters principales, al definir el argumento 'k=3'.*

```
[[2]]
 Connecticut         Idaho       Indiana        Kansas      Kentucky       Montana
           7            12            14            16            17            26
    Nebraska          Ohio Pennsylvania          Utah
          27            35            38            44
```

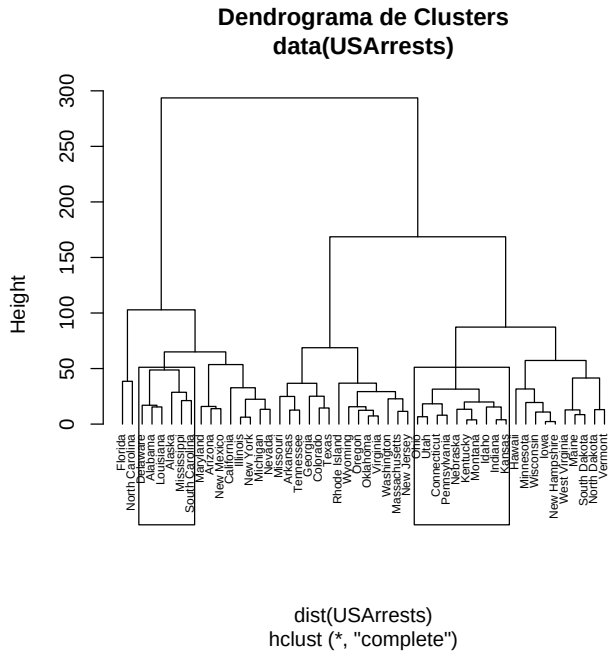

Figura 4.19: *La función 'rect.hclust' También permite identificar en la gráfica clusters particulares indicando sus respectivos números en el argumento 'which'. En la gráfica aparecen identificados con los rectángulos los clusters 2 y 7.*

4.5 Series de tiempo

Series de tiempo pueden ser analizadas en R usando las funciones disponibles en los paquetes base, boot y ts, con las siguientes funciones (Entre paréntesis se indica el nombre del paquete en el cual se haya disponible la correspondiente función, para detalles consultar 'R Documentation'):

- **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)**: Muestrea tiempos de series de tiempos.
- **ts(base)**: Crea objetos tipo series de tiempos.
- **acf(ts)**: Estima la función de autocovarianza o 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)**: Filtramiento lineal de un serie.
- **lag.plot(ts)**: Grafica series de tiempo contra sus versiones rezagadas.
- **lag(ts)**: Rezaga una serie de tiempo.
- **diff.ts(ts)**: Diferencia una serie de tiempo.
- **na.omit.ts(ts)**: Rutinas para manipular valores NA para series de tiempo.
- **spec.ar(ts)**: Ajusta un modelo AR o usa el ajuste existente y estima la densidad espectral de una serie de tiempo.

- **spec.pgram(ts)**: Estima la densidad espectral de una serie de tiempo a partir del periodograma suavizado.
- **spec.taper(ts)**: *taper?* una serie de tiempo.
- **stl(ts)**: Descomposición estacional de una serie de tiempo por LOESS.
- **ts.plot(ts)**: Gráfica series de tiempo sobre el mismo gráfico, tal como plot.ts, con la diferencia de que las series pueden tener bases de tiempo diferentes pero con las mismas frecuencias.
- **ts.union(ts)**: Pega dos o más series de tiempo.
- **plot.acf(ts)**: Grafica funciones de autocovarianza y autocorrelación. Esta función es invocada por defecto or 'acf'.

4.5.1 Función `plot.ts`

Grafica objetos de la clase "ts" o "mts" (series multivariadas).

```
plot(x, y = NULL, type = "l", xlim = NULL, ylim = NULL,
xlab = "Time", ylab, log = "",
col = par("col"), bg = NA, pch = par("pch"),
cex = par("cex"), lty = par("lty"), lwd =
par("lwd"), axes = TRUE, frame.plot = axes,
ann = par("ann"), main = NULL, plot.type =
c("multiple", "single"), xy.labels = n <= 150,
xy.lines = do.lab, panel=lines, ...)

lines(x, ...)
```

Argumentos:

- x,y: Objetos series de tiempo, generalmente de la clase "ts".
- type: Tipo de gráfico (ver 'plot'): Cuando y es especificado, el valor por defecto depende de lo que se especifique en 'xy.labels'.

- plot.type: Para series de tiempo multivariadas, ¿Deberían graficarse separadamente las series o en un mismo gráfico?.
- xy.labels: Argumento lógico, indica si etiquetas 'text()' deberían usarse para un gráfico x-y.
- xy.lines: Argumento lógico, indica si 'lines' debe ser dibujado para un gráfico x-y. Por defecto es TRUE cuando las etiquetas también son dibujadas (argumento anterior).
- panel: Es una función 'function(x, col, bg, pch, type, ...)' que da la acción a ser ejecutadas en cada panel de la presentación para 'plot.type="multiple" '. Por defecto es 'lines'.
- ...: Argumentos gráficos adicionales pueden ser especificados ver 'plot', 'plot.default' y 'par'.

Detalles:

- Con un sólo argumento principal, esta función crea gráficos de series de tiempo, para series multivariadas de dos clases depende de 'plot.type'.
- si y es especificado, tanto x como y deben ser univariados, y será dibujado un gráfico de dispersión de $y \sim x$, mejorado usando 'text' si 'xy.labels' es 'TRUE' o 'caracter', y 'lines' si 'xy.lines' es 'TRUE.

Ejemplo: Creación de una serie y su graficación:

```
#Generación de una serie multivariada de
#3 series con 100 observaciones'ts'

#que inician en enero de 1984,
#con frecuencia mensual:

z <- ts(matrix(rnorm(300), 100, 3), start=c(1984, 1),
frequency=12)

#las tres series presentadas por separados:
```

```
plot(z,main="Serie multivariada simulada a partir
de una N(0,1)\nFrecuencia
mensual",xlab="tiempo",cex.main=0.9)

#las 3 series presentadas superpuestas

plot(z, plot.type="single",lty=1:3,
main="Serie multivariada simulada a partir de una
N(0,1)\nFrecuencia mensual",xlab="tiempo",cex.main=0.9)

legend(1990,-2,c("serie 1","serie 2","serie 3"),
lty=1:3,cex=0.9)
```

4.5.2 Funciones `acf, pacf, ccf`

La función 'acf' calcula y grafica por defecto estimaciones de la función de autocovarianza o de autocorrelación. 'pacf' se usa para las autocorrelaciones parciales y 'ccf' para la autocorrelación o autocovarianza cruzada de dos series univariadas.

```
acf(x, lag.max = NULL,
type = c("correlation", "covariance", "partial"),
plot = TRUE,
na.action, demean = TRUE, ...)

pacf(x, lag.max = NULL, plot = TRUE, na.action, ...)

ccf(x, y, lag.max = NULL,
type =c("correlation","covariance"), plot = TRUE,
na.action,...)
```

Argumentos:

- x, y: Un objeto serie de tiempo univariada o multivariada (excepto para ccf') o un vector o matriz numéricos.

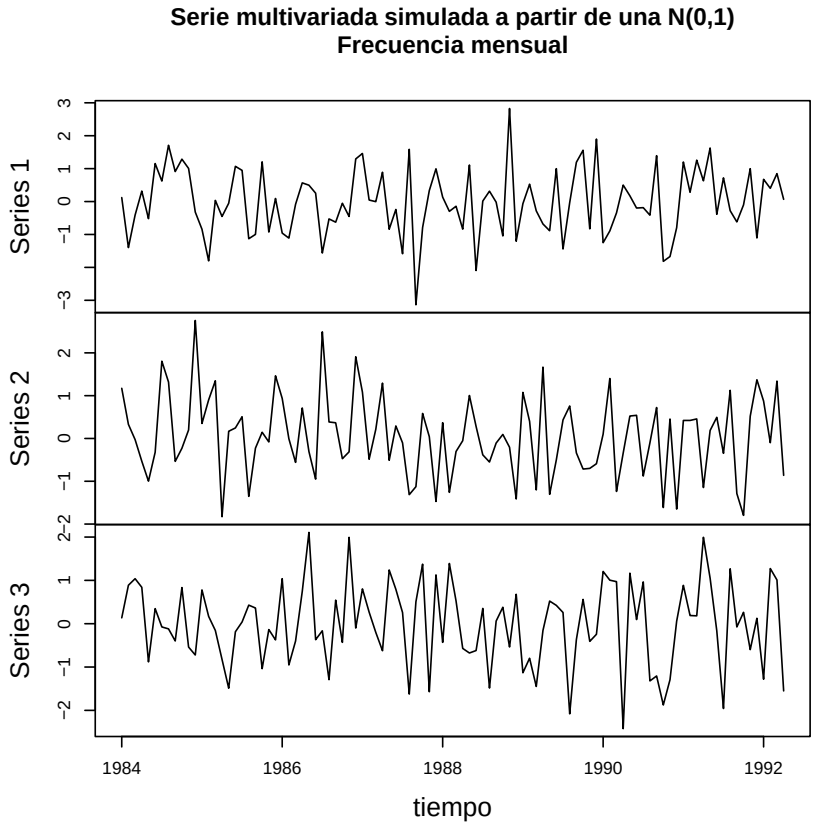

Figura 4.20: *Las series componentes de una serie multivariadas, graficadas con plot.ts o simplemente plot de un objeto de clase "mts"*

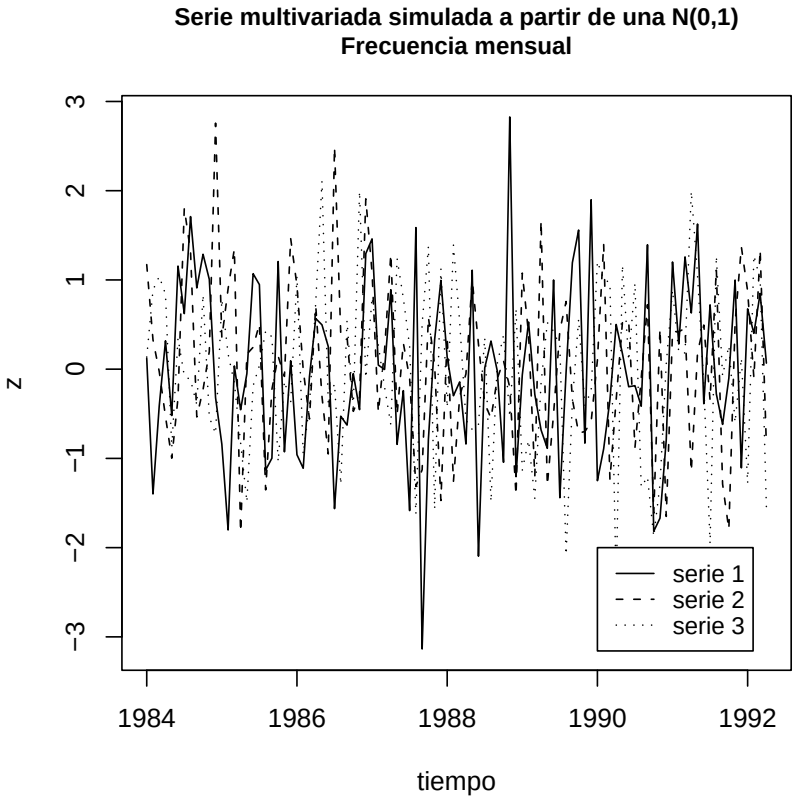

Figura 4.21: *Las series componentes de una serie multivariadas, graficadas con plot.ts en forma superpuesta utilizando el argumento plot.type="single" en 'plot.ts'*

- lag.max: Rezago máximo en el cual calcular el acf. Por defecto es $10 \times log10(N)$ con N siendo el número de observaciones.

- type: Cadena caracter que da el tipo de acf a ser calculado: "correlation" (por defecto), "covariance" o "partial".

- plot: Argumento lógico. Si es TRUE el acf es graficado.

- na.action: Función a ser llamada para el manejo de valores faltantes.

- demean: Argumento lógico. ¿Deberían las covarianzas basarse en las medias muestrales?

- ...: Pueden especificarse argumentos adicionales en 'plot.acf'.

Detalles:

- Para 'type' = "correlation" and "covariance", las estimaciones están basadas en la covarianza muestral.

- El coeficiente de autocorrelación parcial es estimado ajustando modelos autorregresivos de órdenes sucesivamente superiores hasta 'lag.max'.

Valor: Un objeto de clase "acf", que da una lista que indica:

- lag: Un arreglo tridimensional que contiene los rezagos en los cuales el acf es estimado.

- acf: Un arreglo con las mismas dimensiones que 'lag' que contiene el acf estimado.

- type: El tipo de correlación.

- n.used: El número de observaciones en la serie de tiempo.

- series: El nombre de la serie 'x'.

- snames: Los nombres de las series para una serie multivariada.

El resultado de esta función no es visible si 'plot' es 'TRUE', es decir, el resultado es un gráfico. si se desea obtener tanto el resultado como el gráfico, asignar la función a un objeto y luego llamarlo. El nivel de confianza puede variarse especificando '$ci = (1 - \alpha)$', por defecto es del 95%.

Autores:

- Original: Paul Gilbert, Martyn Plummer.
- Extensive modifications and univariate case of 'pacf' by B.D. Ripley.

Ver también: 'plot.acf'

Ejemplo 1: Este ejemplo consiste de una serie AR(1) con $\phi = 0.8$ simulada. Aunque aplicar directamente las funciones acf o pacf sobre un objeto 'ts' produce el correspondiente gráfico, En este ejemplo no se hará así para poder manipular las etiquetas de los ticks en el eje x que indican los rezagos:

```
#Generación de las observaciones
#inicio de rezago con valor de 0

#Se generan 350 observaciones

library(ts)

zt.1<-0
zt<-rep(0,350)
for(i in 1:350){
zt[i]<-0.8*zt.1+rnorm(1)
zt.1<-zt[i]
}

#Los datos generados son convertidos
#a un objeto 'ts'
#pero se conservan sólo los
#últim0s 300 para tener
```

```
#una serie más estable.
#La serie inicia en enero de 1978

#y la frecuencia es mensual:


zt<-ts(zt[51:350],start=c(1978,1),frequency=12)

plot(zt, main=expression(paste("Proceso AR(1) simulado",sep=" ",
phi==0.8)),
xlab="Tiempo")

#crear un objeto tipo 'acf':

q<-acf(zt,plot=FALSE)

#verificar el número de rezagos:
length(q$lag)
[1] 25

#Graficar el acf:

a<-layout(matrix(c(1,2),ncol=1,nrow=2,byrow=T),
widths=10,heights=c(7,7),respect=T)

par(mar=c(5,4,3,1))
plot(q,main="",sub="(a)",cex.sub=0.8,
cex.axis=0.7,cex.lab=0.7)


par(mar=c(5,4,2,1))
plot(q,xaxt='n',main="",sub="(b)",
cex.sub=0.8,cex.axis=0.7,cex.lab=0.7)
axis(1,at=q$lag,labels=as.character(c(0:24)),cex.axis=0.7)

par(oma=c(1,1,2,1),cex=0.8,font=2)
mtext(outer=T,expression(paste("Proceso AR(1)
simulado",sep="   ",phi==0.8)),side=3)
```

```
#Crear un objeto tipo ‘pacf’:

p<-pacf(zt,plot=FALSE)
length(p$lag)
[1] 24


#Graficar el pacf:

a<-layout(matrix(c(1,2),ncol=1,nrow=2,byrow=T),
widths=10,heights=c(7,7),respect=T)

par(mar=c(5,4,3,1)) plot(p,main="",sub="(a)",
cex.sub=0.8,cex.axis=0.7,cex.lab=0.7)

par(mar=c(5,4,2,1))
plot(p,xaxt='n',main="",sub="(b)",cex.sub=0.8,
cex.axis=0.7,cex.lab=0.7)
axis(1,at=p$lag,labels=as.character(c(1:24)),cex.axis=0.7)

par(oma=c(1,1,2,1),cex=0.8,font=2)
mtext(outer=T,expression(paste("Proceso AR(1)
simulado",sep="   ",phi==0.8)),side=3)
```

Ejemplo 2: Un proceso ARIMA:

```
#Generaci\'on de la serie, datos
#mensuales a partir de julio de 1982:

gnp <- ts(cumsum(1 + round(rnorm(100), 2)),
start = c(1982, 7), frequency = 12)

#Diferenciación de la serie
#(primera diferencia):
```

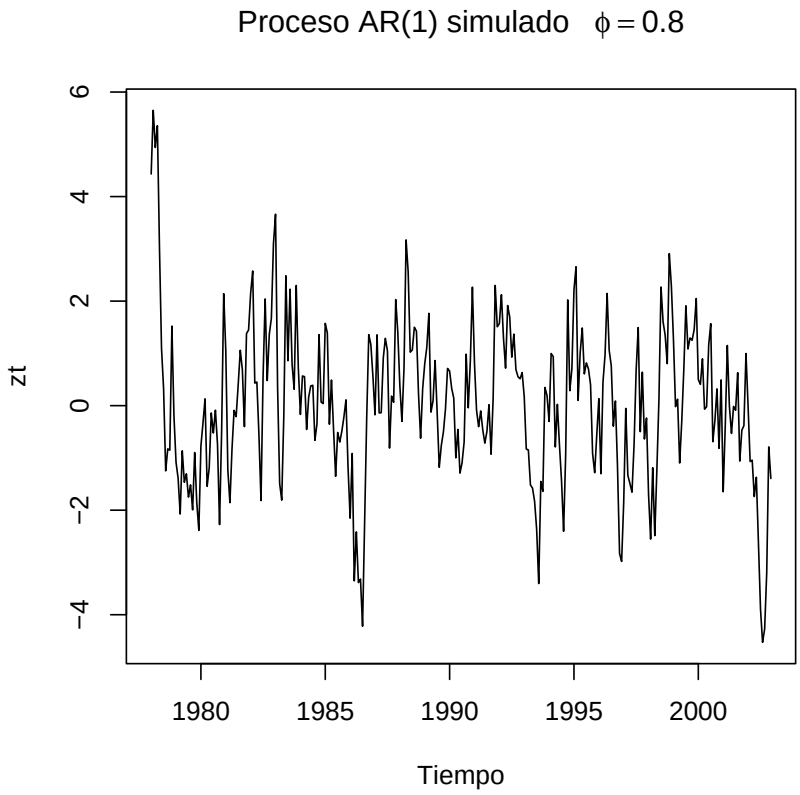

Figura 4.22: *Proceso AR(1) simulado con* $\phi = 0.8$ *con los errores distribuidos N(0,1).*

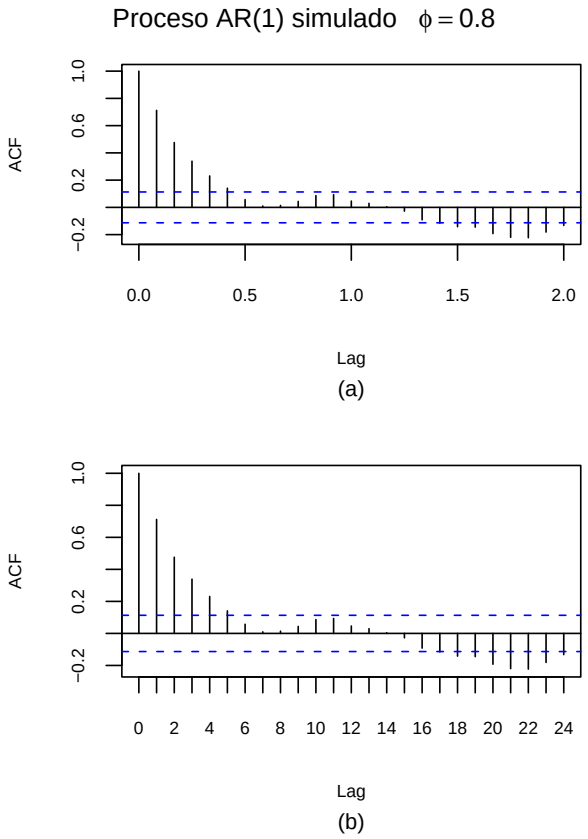

Figura 4.23: *(a) y (b) corresponden al ACF de la misma serie, pero observe las etiquetas de los rezagos en el eje horizontal, en (a) los valores señalados corresponden a la posición en el eje de las observaciones (y no al valor del rezago), como múltiplos de* $1/12$ *de año, dado que el objeto zt fue creado como una serie de observaciones mensuales, este es el resultado por defecto de la función 'plot.acf' cuando se aplica a un objeto de clase "ts" de frecuencia inferior a un año, pero esto no sucede cuando se aplica a un objeto vectorial simple o a un objeto clase "ts" de frecuencia anual. En (b) se ha corregido lo anterior.*

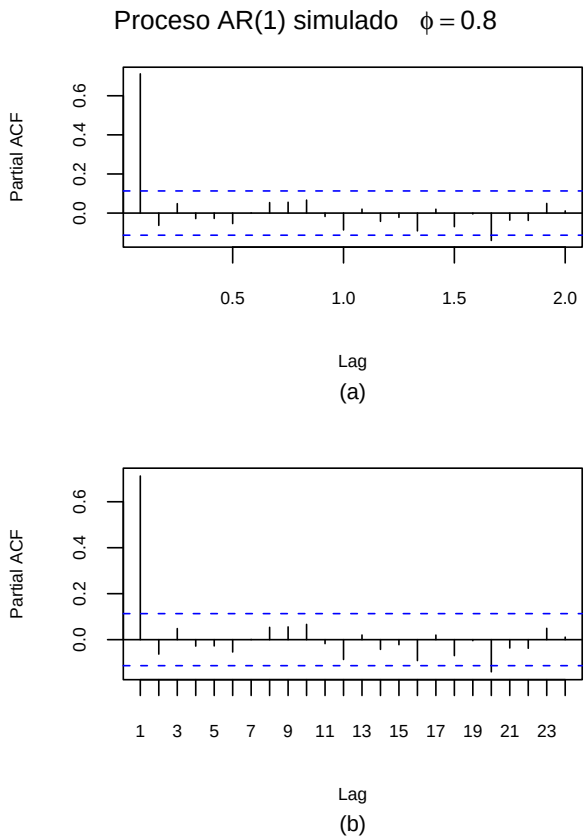

Figura 4.24: *(a) y (b) corresponden al PACF de la misma serie, pero observe las diferencias en las etiquetas de los rezagos como se vió en el caso del ACF.*

```
d<-diff.ts(gnp,lag=1,differences=1)

#Presentación de gráficas
#Serie original:

a<-layout(matrix(c(1,2),ncol=1,nrow=2,byrow=T),
widths=10,heights=c(7,7),respect=T)

par(mar=c(5,4,3,1))
plot(gnp,main="Proceso ARIMA
simulado",sub="(a)",cex.main=0.7,
cex.sub=0.8,cex.axis=0.7,cex.lab=0.7,xlab="Tiempo")

par(mar=c(5,4,2,1))
plot(d,main="Serie
diferenciada",sub="(b)",cex.main=0.7,
cex.sub=0.8,cex.axis=0.7,cex.lab=0.7,xlab="Tiempo")

acf.orig<-acf(gnp,plot=FALSE)
length(acf.orig$lag)
[1] 21

pacf.orig<-pacf(gnp,plot=FALSE)
length(pacf.orig$lag)
[1] 20

a<-layout(matrix(c(1,2),ncol=1,nrow=2,byrow=T),
widths=10,heights=c(7,7),respect=T)
par(mar=c(5,4,3,1))
plot(acf.orig,xaxt='n',main="",sub="(a)",cex.sub=0.8,
cex.axis=0.7,cex.lab=0.7)
axis(1,at=acf.orig$lag,labels=as.character(c(0:20)),
cex.axis=0.7)

par(mar=c(5,4,2,1))
plot(pacf.orig,xaxt='n',main="",sub="(b)",cex.sub=0.8,
cex.axis=0.7,cex.lab=0.7)
axis(1,at=pacf.orig$lag,labels=as.character(c(1:20)),
```

```
cex.axis=0.7)

par(oma=c(1,1,2,1),cex=0.8,font=2)
mtext(outer=T,"Proceso ARIMA simulado",side=3)


#Presentación de gráficas
#serie diferenciada:

acf.d<-acf(d,plot=FALSE)
length(acf.d$lag)
[1] 20

pacf.d<-pacf(d,plot=FALSE)
length(pacf.d$lag)
[1] 19

a<-layout(matrix(c(1,2),ncol=1,nrow=2,byrow=T),
widths=10,heights=c(7,7),respect=T)
par(mar=c(5,4,3,1))
plot(acf.d,xaxt='n',main="",sub="(a)",cex.sub=0.8,
cex.axis=0.7,cex.lab=0.7)
axis(1,at=acf.d$lag,labels=as.character(c(0:19)),
cex.axis=0.7)

par(mar=c(5,4,2,1))
plot(pacf.d,xaxt='n',main="",sub="(b)",cex.sub=0.8,
cex.axis=0.7,cex.lab=0.7)
axis(1,at=pacf.d$lag,labels=as.character(c(1:19)),
cex.axis=0.7)

par(oma=c(1,1,2,1),cex=0.8,font=2)
mtext(outer=T,"Proceso ARIMA simulado
Diferenciado",side=3)
```

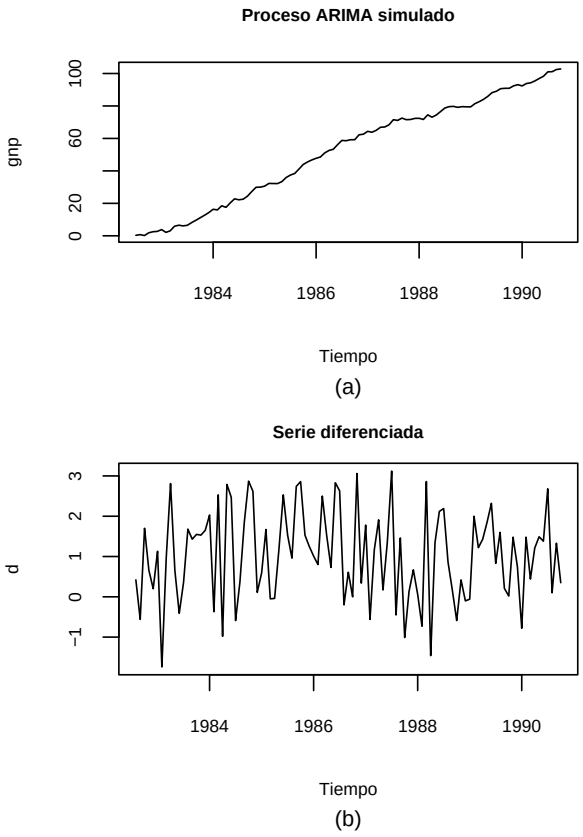

Figura 4.25: *(a) Proceso de caminata aleatoria simulada y (b) la serie diferenciada una vez.*

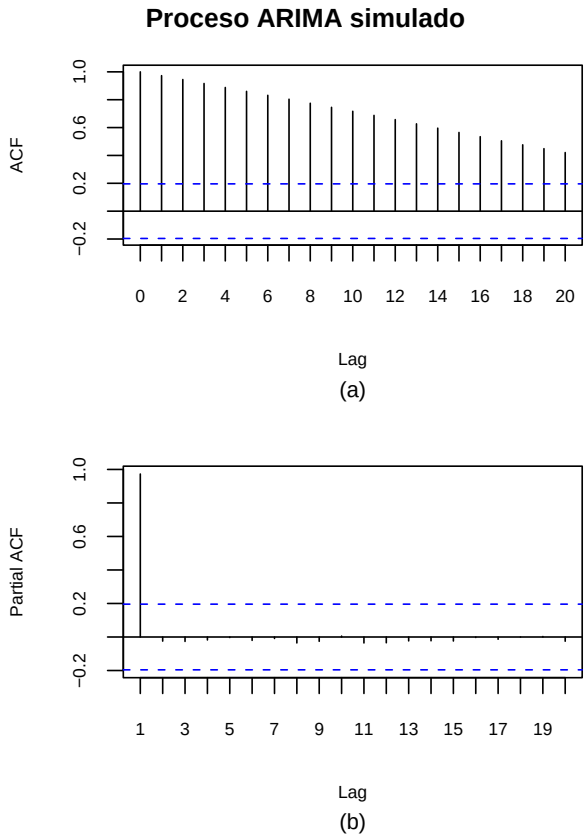

Figura 4.26: *(a) ACF Proceso de caminata aleatoria y (b) PACF correspondiente. Observe que el ACF indica la necesidad de diferenciar la serie.*

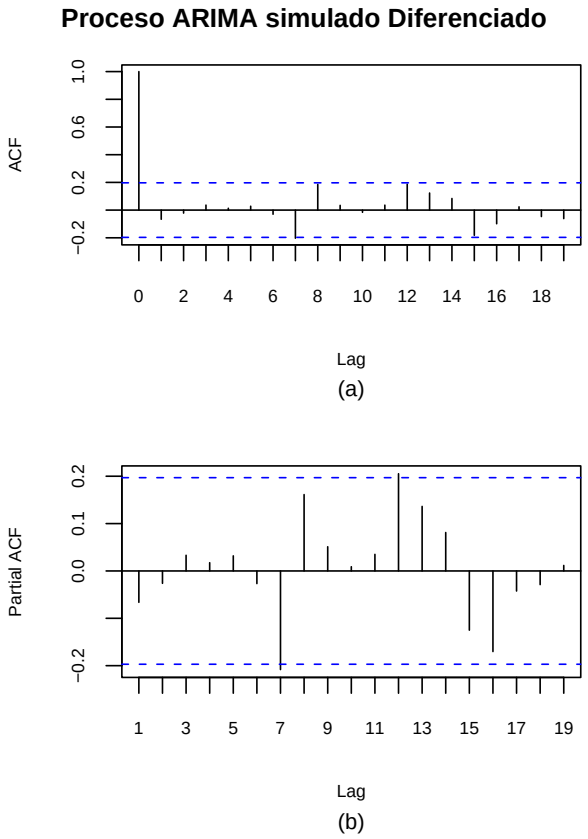

Figura 4.27: *(a) ACF Proceso de caminata aleatoria simulada diferenciada y (b) PACF correspondiente. Observe que el ACF y el PACF indican que la serie diferenciada puede ser un proceso MA?.*

4.5.3 Función `lag.plot`

Permite graficar una serie de tiempo contra sus rezagos, permitiendo visualizar "auto-dependencia" aún cuando las auto-correlaciones *vanish?*.

```
lag.plot(x, lags = 1, layout = NULL, set.lags = 1:lags,
main = NULL, asp = 1,
font.main=par("font.main"), cex.main=par("cex.main"),
diag = TRUE, diag.col="gray",
type="p", oma =NULL, ask =NULL, do.lines = n <= 150,
labels = do.lines, ...)
```

Argumentos:

- x: Serie de tiempo (univariada o multivariada)
- lags: El número de rezagos deseados a graficar, ver 'set.lags'.
- layout: El diseño para presentación de gráficos múltiples, básicamente el argumento 'mfrow' de 'par' de modo que todos los gráficos queden en una misma página.
- set.lags: Vector de enteros positivos que permite especificar el conjunto de rezagos usados; por defecto corresponde a '1:lags'.
- main: Título de encabezado principal a ser usado en la parte superior de cada página.
- asp: Razón de aspecto a ser fijado, ver 'plot.default'.
- font.main, cex.main: Atributos para los caracteres de título, ver 'par()'.
- diag: Argumento lógico para indicar si se traza la diagonal $x = y$.
- diag.col: Color a ser usado para la diagonal.
- type: Tipo de gráfico a ser usado. Ver 'plot.ts'.
- oma: Márgenes externas, ver 'par'.

- ask: Argumento lógico; si es TRUE el usuario es preguntado antes de que una nueva página sea iniciada.
- do.lines: Argumento lógico que indica si líneas deberán ser dibujadas.
- labels: Argumento lógico que indica si deberán usarse etiquetas.
- ...: Argumentos adicionales para 'plot.ts'.

Autor: Martin Maechler

Ejemplo: Para la serie AR(1) simulada previamente, construir los gráficos de rezagos k=1 a 5:

```
lag.plot(zt,lags=6,diag=TRUE,diag.col="blue",
do.lines=FALSE,main=expression(paste("Proceso
AR(1) simulado",sep="   ",phi==0.8)),cex.main=0.8 )
```

4.5.4 Función `stl`

Esta función permite obtener la descomposición de una serie en sus componentes estacional, de tendencia e irregular usando LOESS.

```
stl(x, s.window, s.degree = 0, t.window = NULL,
t.degree = 1, l.window = nextodd(period),
l.degree = t.degree, s.jump = ceiling(s.window/10),
t.jump = ceiling(t.window/10), l.jump
= ceiling(l.window/10), robust = FALSE,
inner = if(robust)  1 else 2,
outer = if(robust)
15 else 0, na.action = na.fail)
```

Argumentos:

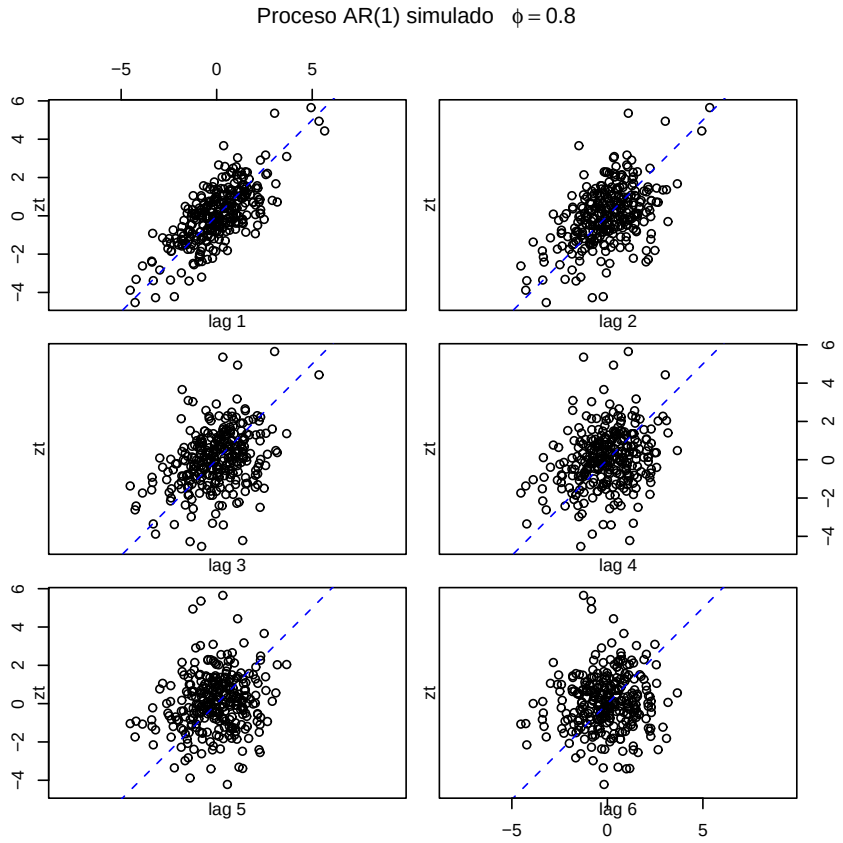

Figura 4.28: *Observe cómo los scatters plot pueden dar indicios acerca de la dependencia entre un serie y sus rezagos, para la determinación de la estructura auto-regresiva. Por ejemplo, entre la serie y el rezago de orden 1 parece existir una asociación importante, la cual se debilita al aumentar en orden del rezago a partir del rezago 2.*

- x: Serie de tiempo univariada a ser descompuesta, debe ser un objeto de clase 'ts' con frecuencia mayor que 1.

- s.window: Puede ser la cadena de caracteres "periodic" o la extensión (en rezagos) del span de la ventana loess para la extracción de la componente estacional, el cual debería ser impar. Este no tiene especificación por defecto.

- s.degree: Grado del polinomio localmente ajustado en la extracción estacional, Debe ser 0 ó 1.

- t.window: El span (en rezagos) de la ventana loess para la extracción de la componente de tendencia, el cual debe ser impar. Si es 'NULL', valor por defecto, '$nextodd(ceiling((1.5 * period)/(1 - (1.5/s.window))))$' es tomado.

  t.degree: El grado del polinomio localmente ajustado en la extracción de tendencia. Debe ser 0 ó 1.

- l.window: El span (en rezagos) de la ventana loess para el *low-pass filter* usado para cada subserie. Por defecto está ajustado al entero impar más pequeño mayor o igual a 'frequency(x)' el cual es recomendado dado que previene competición entre las componentes de tendencia y estacionalidad. Si no es un entero impar su valor dado es incrementado al impar siguiente.

- l.degree: El grado del polinomio localmente ajustado para las sub series *low-pass filter*. Debe ser 0 ó 1.

- s.jump, t.jump, l.jump: Enteros de al menos 1 para incrementar la rapidez del respectivo suavizador. La interpolación lineal sucede entre cada valor '*.jump' iésimo

- robust: Argumento lógico para indicar si un ajuste robusto será usado en el procedimiento 'loess'.

- inner: Entero que da el número de iteraciones 'inner' (*backfitting*). Usualmente con dos iteraciones es suficiente.

- outer: Entero que da el número de iteraciones 'outer' robustas.

- na.action: Acción a seguir con valores faltantes.

Detalles: La componente estacional es hallada por suavizamiento loess de las subseries estacionales (Las series de todos los valores de los eneros, etc...); Si 's.window ="periodic" ' el suavizamiento es reemplazado tomando la media. Los valores estacionales son removidos, y el suavizado restante para hallar la tendencia. El nivel total es removido de la componente estacionan y agregado a la componente de tendencia. Este proceso es iterado unas pocas veces. la componente 'remainder' constituye los residuales del ajuste de estacionalidad más tendencia.

Valor: 'stl' devuelve un objeto de clase "stl" con las siguientes componentes:

- time.series: Una serie de tiempo múltiple con columnas 'seasonal', 'trend' y 'remainder'.
- weights: Pesos finales robustos (todos si el ajuste no es hecho robustamente).
- call: *¿the matched call?*.
- win: Vector de enteros (de longitud 3) con los spans usados por los suavizadores "s","t" e "l".
- deg: Vector de enteros (de longitud 3) con los grados de los polinomios para los suavizadores.
- jump: Vector de enteros (de longitud 3) con los "jumps" (skips) usados por los suavizadores.
- ni: Número de iteraciones internas (inner iterations)
- no: Número de iteraciones robustas externas.

Nota: Esta función es similar a la función 'stl' de S-PLUS, pero difiere en que la componente 'remainder' del S-PLUS es la suma de las series 'trend' y 'remainder' de la función de R.

Autor: B.D. Ripley; Fortran code by Cleveland et al. (1990) from 'netlib'.

Referencias: R. B. Cleveland, W. S. Cleveland, J.E. McRae, and I. Terpenning (1990) STL: A Seasonal-Trend Decomposition Procedure Based on Loess. Journal of Official Statistics, 6, 3-73.

Ejemplo: El siguiente ejemplo está disponible en R:

```
library(ts)
data(co2)
plot(stllc <- stl(log(co2), s.window=21),
main="Descomposición de
una serie",cex.main=0.9)

summary(stllc)
```

```
 Call:
 stl(x = log(co2), s.window = 21)

 Time.series components:
    seasonal                 trend            remainder
 Min.   :-9.939103e-03   Min.   :5.753541   Min.   :-2.255405e-03
 1st Qu.:-4.536535e-03   1st Qu.:5.778556   1st Qu.:-4.586796e-04
 Median : 8.777611e-04   Median :5.815125   Median :-8.867395e-06
 Mean   :-1.304469e-06   Mean   :5.819267   Mean   :-1.965549e-06
 3rd Qu.: 4.997747e-03   3rd Qu.:5.859806   3rd Qu.: 4.023465e-04
 Max.   : 9.114691e-03   Max.   :5.898750   Max.   : 1.939626e-03
 IQR:
     STL.seasonal STL.trend STL.remainder data
     0.009534     0.081250  0.000861      0.079370
   %  12.0        102.4       1.1         100.0

 Weights: all == 1

 Other components: List of 5
 $ win  : Named num [1:3] 21 21 13
 $ deg  : Named int [1:3] 0 1 1
 $ jump : Named num [1:3] 3 3 2
 $ inner: int 2
 $ outer: int 0
```

```
#Extraer las 3 series generadas por stl
#en el objeto stllc, formando una matriz
```

```
#de tres columnas:

sts<- .Alias(stllc$time.series)

#Serie del resto:

resto<-sts[,"remainder"]

acf.resto<-acf(resto,plot=FALSE)
length(acf.resto$lag)
[1] 27

pacf.resto<-pacf(resto,plot=FALSE)
length(pacf.resto$lag)
[1] 26

a<-layout(matrix(c(1,2,3),ncol=1,nrow=3,byrow=T),
widths=10,heights=c(3.5,3.5,3.5),respect=T)

par(mar=c(5,4,3,1))
plot(resto,main="",xlab="Tiempo",sub="(a)",cex.sub=1,
cex.axis=0.7,cex.lab=1)

par(mar=c(5,4,2,1))
plot(acf.resto,xaxt='n',main="",sub="(b)",cex.sub=1,
cex.axis=0.7,cex.lab=1)
axis(1,at=acf.resto$lag,labels=as.character(c(0:26)),
cex.axis=0.7)

par(mar=c(5,4,2,1))
plot(pacf.resto,xaxt='n',main="",sub="(c)",cex.sub=1,
cex.axis=0.7,cex.lab=1)
axis(1,at=pacf.resto$lag,labels=as.character(c(1:26)),
cex.axis=0.7)

par(oma=c(1,1,2,1),cex=0.8,font=2)
mtext(outer=T,"Componente estacionaria serie
```

```
data(co2)",side=3)
```

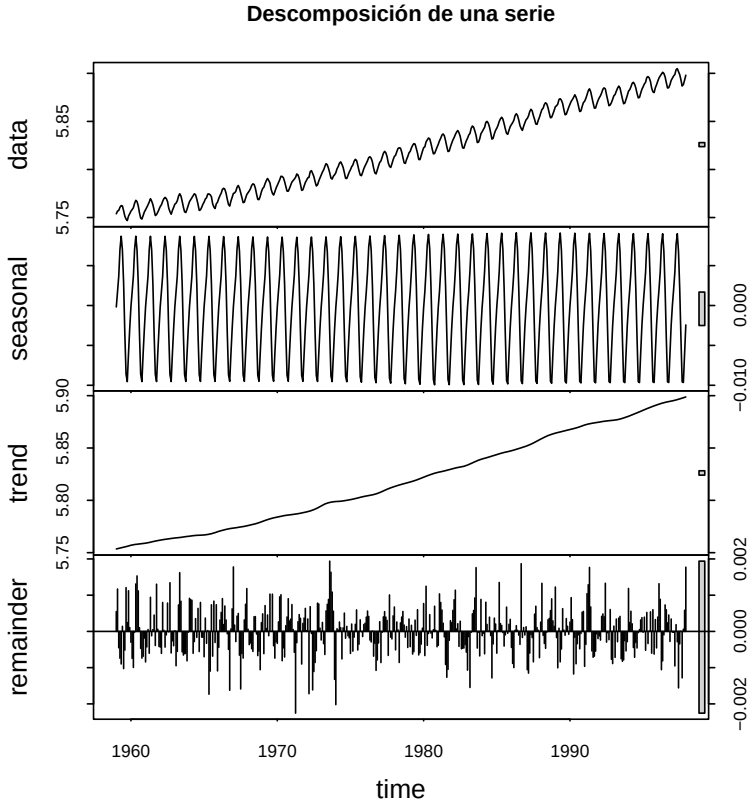

Figura 4.29: *El gráfico presenta la serie del log de las observaciones disponibles en data(co2) y su descomposición en las componentes estacional, de tendencia y estacionaria, usando LOESS mediante la función 'stl'.*

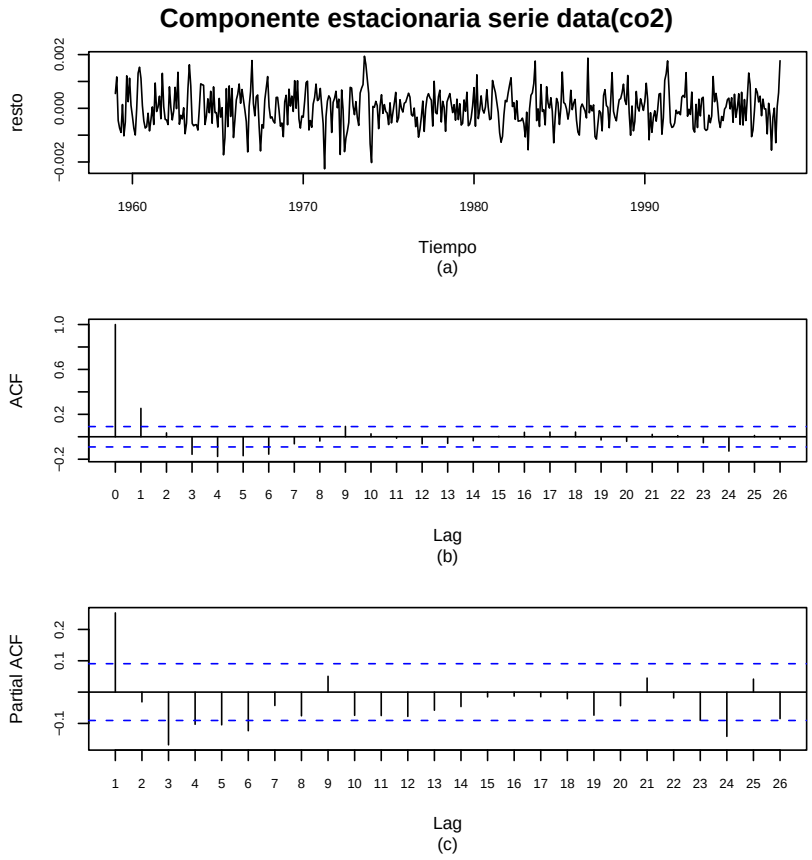

Figura 4.30: *El gráfico presenta la componente estacionaria del log de las observaciones en data(co2) así como el ACF y PACF. Parece que un modelo tipo ARMA ajusta esta serie.*

Capítulo 5

Gráficos Condicionales

5.1 Gráficas Trellis

Las gráficas Trellis representan un paso de mayor importancia en gráficas estadísticas. Ellos proporcionan una forma general a las grficas múltiples y condicionales.

La función coplot permite obtener este tipo de gráficos, sus argumentos son:

```
coplot(formula, data, given.values, panel = points, rows,
columns,
show.given = TRUE, col = par("fg"), pch = par("pch"),
xlab = paste("Given :", a.name),
ylab = paste("Given :", b.name),
number = 6, overlap = 0.5, ...)
co.intervals(x, number = 6, overlap = 0.5)
```

- formula: Para definir la forma del gráfico condicional como $y \sim x|a$, indicando que se graficará y versus x condicionados a la variable a. Si la fórmula es de la forma $y \sim x|a * b$ entonces los gráficos de y versus x serán condicionados sobre las variables a y b.x y y deben ser numéricas pero a y b pueden no serlo.

- data: Un marco de datos que contiene los valores para cualquiera de las variables en la fórmula, aunque por defecto coplot usa los datos del ambiente en el cual ha sido invocado.

- given.values: Corresponde a un valor o lista de dos valores los cuales determinan cómo los condicionamientos sobre a y b van a darse. Si sólo se condiciona sobre una variable, entonces este argumento generalmente corresponde a una matriz con dos columnas, con cada fila dando un intervalo sobre el cual se condiciona, pero también puede ser un vector de números o un conjunto de niveles de un factor cuando la variable sobre la que se condiciona es un factor. En caso de sólo una variable condicionante, el resultado de co.intervals(..) puede ser usado directamente como el valor de este argumento.

- panel: Es una función $function(x, y, col, pch, ...)$ que realiza la acción a ser ejecutada en cada panel del gráfico. Por defecto es "points".

- rows: Los paneles del gráfico son puestos en un arreglo de filas por columnas. Este argumento da el número de filas en el arreglo.

- columns: Corresponde al número de columnas en el arreglo de paneles.

- show.given: Es un argumento lógico, que puede ser de longitud dos cuando se tienen dos variables condicionantes. Por defecto (TRUE) los gráficos se presentan para las correspondientes variables condicionantes

- col: Para especificar un vector de colores a ser usados para los puntos. Si es muy corto los valores son reciclados.

- pch: Para especificar un vector de símbolos de graficación.

- xlab, ylab: Para etiquetar a la primera y segunda variable condicionante respectivamente.

- number: Número entero, para indicar el número de intervalos de la variable condicionante. Puede ser de longitud dos para la dirección x y y.

- overlap: Un valor numérico < 1, con el que se especifica la fracción de superposición de los intervalos de las variables condicionantes, puede ser de longitud 2 para la dirección x e y. Un valor negativo produce huecos entre tramos de datos.

Ejemplo 1:

```
datos.huevos<-matrix(scan('c:/graficos/huevos.dat'),ncol=7,
byrow=T)
peso.inicial<-datos.huevos[,3]
peso.final<-datos.huevos[,6]
dureza<-datos.huevos[,7]
volumen<-datos.huevos[,4]
tiempo<-datos.huevos[,5]
coplot(peso.inicial ~ peso.final | volumen)
```

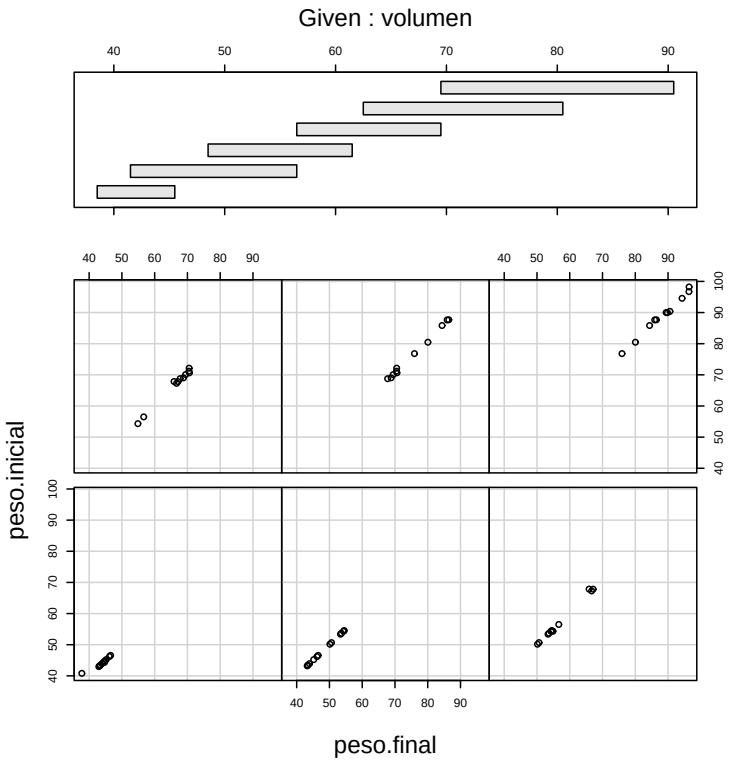

Figura 5.1: *Este gráfico señala la relación entre los pesos iniciales y finales según niveles del volumen del huevo. Debe mirarse por fila (desde arriba) de derecha a izquierda*

Ejemplo 2:

```
given.volumen<-co.intervals(volumen,number=4,overlap=0)
coplot(peso.inicial ~ peso.final | volumen,
given.values=given.volumen)
```

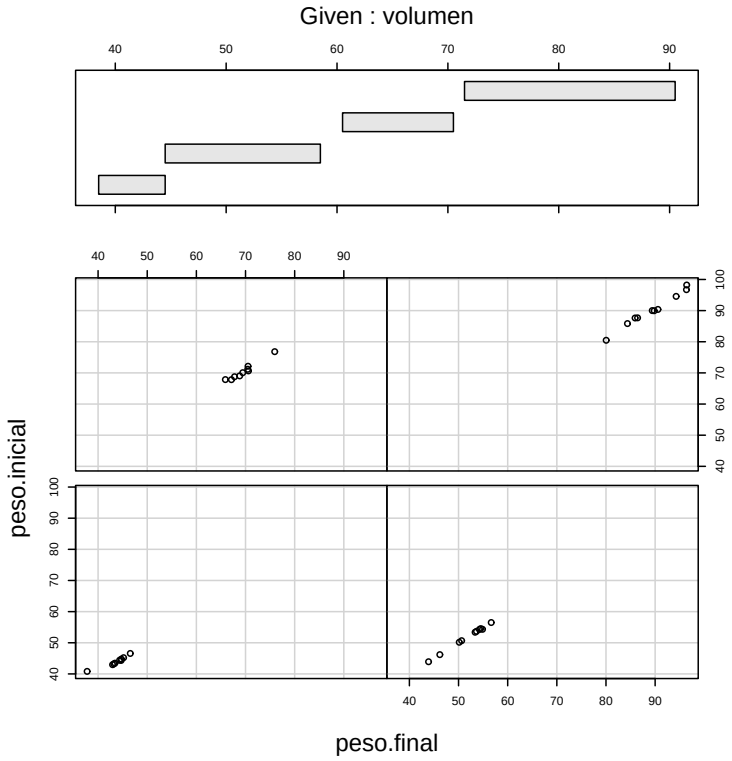

Figura 5.2: *El gráfico anterior modificado: cuatro intervalos de volumen sin superposición.*

ejemplo 3:

```
pfv.td<-peso.final ~ volumen | tiempo*as.factor(dureza)
coplot(pfv.td)
```

ejemplo 4:

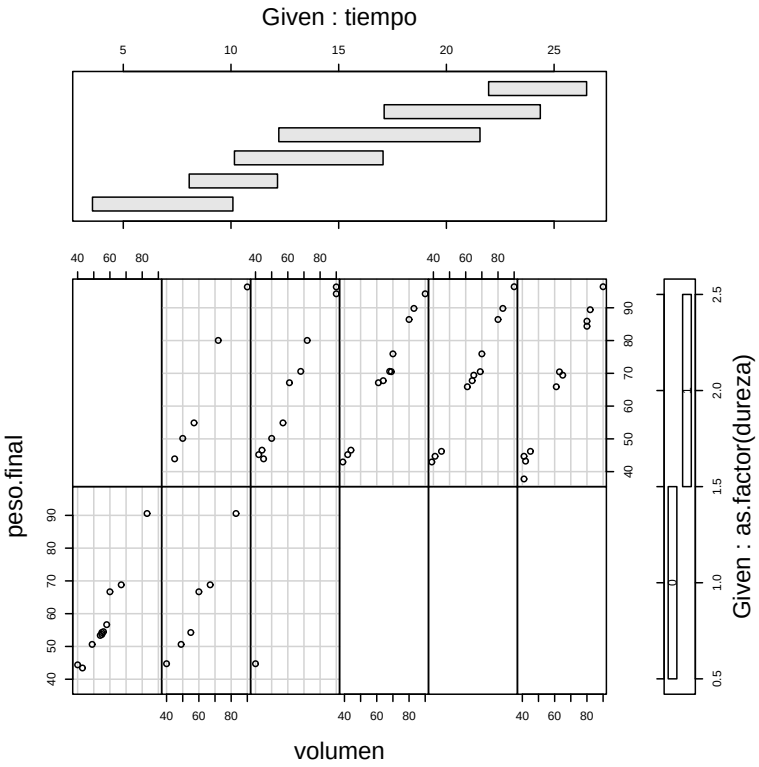

Figura 5.3: *Hay paneles vacíos indicando que con más de 15 minutos los huevos endurecen y que abajo de 6 min aproximadamente los huevos quedan blandos.*

```
datos.icfes<-read.table('c:/graficos/icfes.dat')
sexo<-as.factor(datos.icfes[,1])
conoc.mat<-datos.icfes[,10]
apt.mat<-datos.icfes[,9]
electiva<-as.factor(datos.icfes[,12])
apt.verbal<-datos.icfes[,7]
fisica<-datos.icfes[,5]
coplot(conoc.mat ~ apt.mat | sexo*electiva,
panel=function(x,y,...) panel.smooth(x,y, span= .8, ...))
```

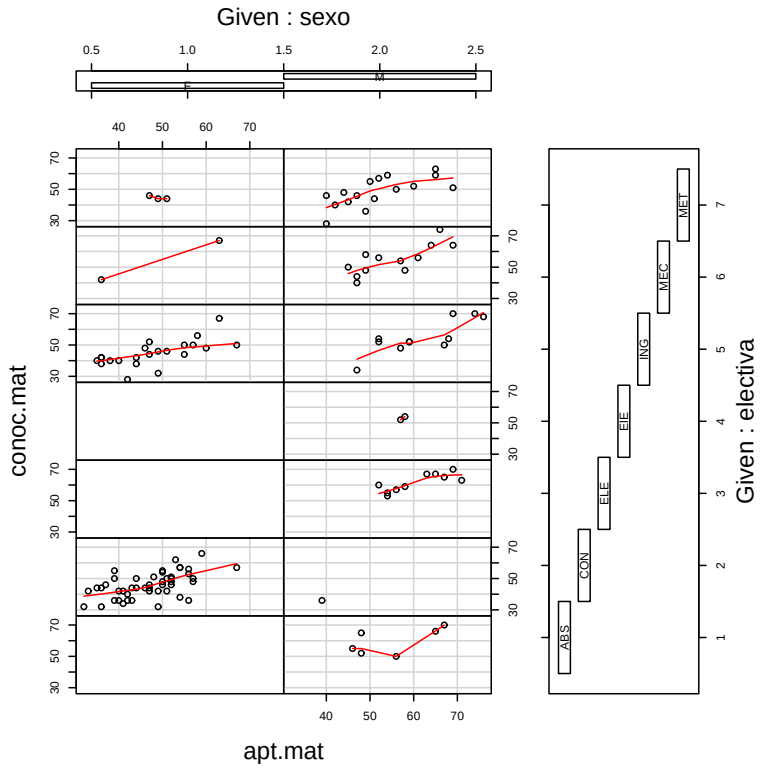

Figura 5.4: *Hay paneles vacíos indicando que las mujeres no tomaron las electivas correspondientes. En cuanto a la influencia del sexo y la electiva sobre los resultados en matemática no parecen ser determinantes.*

ejemplo 5:

```
aux<- fisica ~ conoc.mat | apt.verbal*apt.mat
coplot(aux,number=c(3,4),overlap=c(0,0.1))
```

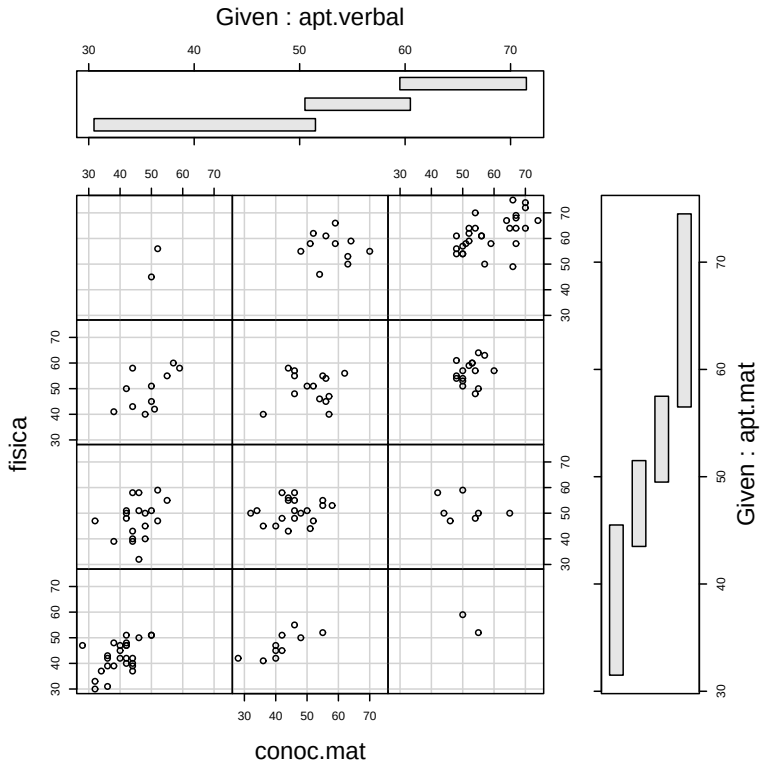

Figura 5.5: *A mayor aptitud matemática y verbal mejores resultados en física y en matemáticas. también quien tiene buenos resultados en matemáticas logra buenos resultados en física.*

Ejemplo 6:

```
coplot(aux,number=c(3,4),show.given=c(F,F),overlap=c(0,0.1))
```

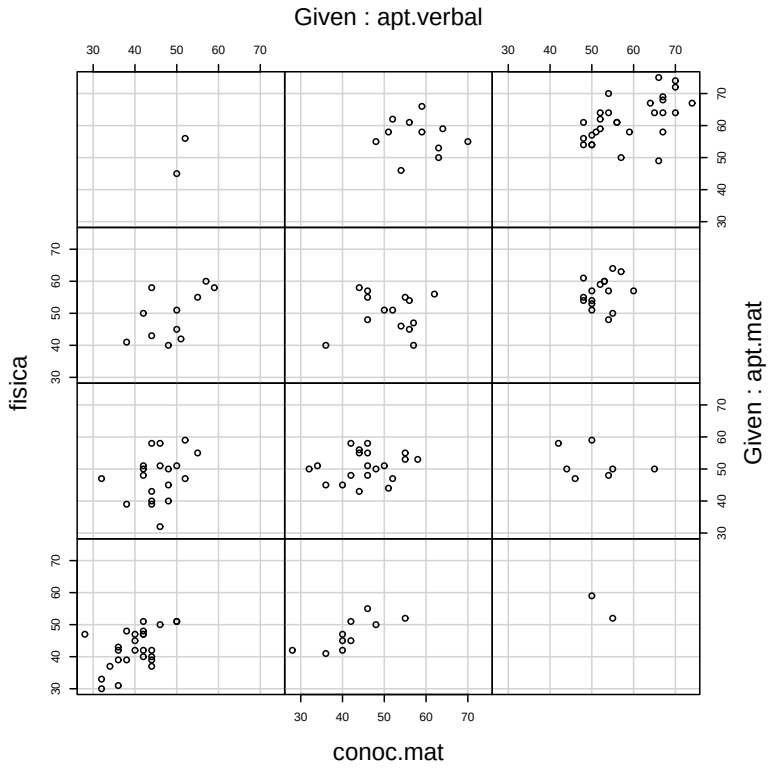

Figura 5.6: *El mismo ejemplo en la figura 5.5 pero sin los paneles correspondientes a los niveles de las variables condicionantes.*

5.2 Gráfico de Perfil

Un gráfico de perfil es una generalización del gráfico bidimensional, utilizando una serie de ejes verticales presentados consecutivamente a lo largo de la base (el eje x) del gráfico. Cualquier nḿero de variables respuesta puede ser tenido en cuenta con sus respectivas escalas de medición. Las variables respuesta se organizan a lo largo de la base del gráfico, parecido a datos discretos. El conjunto de respuestas de cada sujeto es graficada sobre los correspondientes ejes verticales, conectando los puntos con líneas entre sí. Cada línea define el "perfil" de respuestas de un sujeto. Se puede utilizar colores o diferentes tipos de líneas para poder discriminar entre poblaciones o tratamientos.

Supongamos que una evaluación consiste de cuatro preguntas, cada una calificada de 0 a 5. 50 personas presentan la evaluación y los resultados obtenidos mediante la siguiente simulación en la cual las calificaciones a las respuestas a cada pregunta corresponden a los datos de las columnas en su orden:

```
datos<-matrix(rep(0,200),ncol=4)
calificacion<-function(n,mu,sigma){
datos<-abs(rnorm(1000,mu,sigma))
datos<-(datos)
datos<-datos[datos<=5]
datos<-sample(datos,n,replace=FALSE)
datos
}
> datos[,1]<-calificacion(50,3,1)
> datos[,2]<-calificacion(50,2,2)
> datos[,3]<-calificacion(50,4,3)
> datos[,4]<-calificacion(50,1,2)
```

La siguiente función permite crear el respectivo gráfico de perfiles, donde los argumentos datos y calmax son respectivamente la matriz que contiene las calificaciones por pregunta y el valor máximo de la escala de calificación:

```
perfil<-function(datos,calmax){
```

```
n<-ncol(datos)
c<-calmax m<-nrow(datos) for(i in 1:n){
plot(rep(i,m),datos[,i],type='n',xlim=c(1,n),ylim=c(0,c),
xlab="",ylab="",axes=FALSE)
par(new=T)
}
axis(1,1:n,LETTERS[1:n])
axis(2)
title(main="Gr\'afico de
Perfiles",xlab="preguntas",ylab="respuestas")
for(i in 1:n){
for(j in 1:calmax){
points(i,j,pch=19)
}
}
for(i in 1:(n-1)){
segments(rep(i,m),datos[,i],rep(i+1,m),datos[,i+1],col="black")
}
}
perfil(datos,5)
```

El gráfico resultante corresponde a la figura 5.7.

Obsérvese que en la función plot se definió el argumento "axes=FALSE", es decir, se indicó no dibujar los ejes coordenados. Luego se invocó la función axis la cual permite agregar ejes a un gráfico previo. Esta función tiene los siguientes argumentos:

```
axis(side, at, labels=TRUE, ...)
```

- side: Toma el valor de 1 para dibujar un eje horizontal inferior (eje x), 2 para un eje vertical izquierdo (eje y), 3 para un eje horizontal superior y 4 para un eje vertical derecho.

- at: Un vector para indicar los puntos sobre los cuales se colocarán las marcas o ticks del eje.

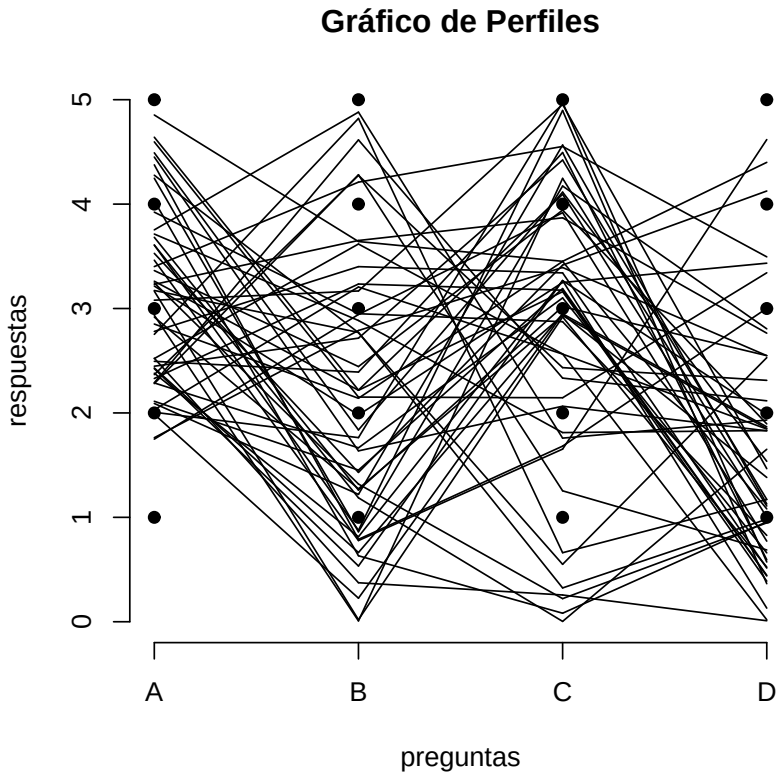

Figura 5.7: *Puede observarse por los perfiles trazados por cada linea, que la pregunta A presenta una calificación entre 2 y más de 4, pero con tendencia por encima de 3; la pregunta B, con calificaciones más dispersas, señala que la mayoría recibe una nota inferior a 3; los perfiles para la pregunta C señalan una nota para la mayoría entre 3 y 5, y la pregunta D presenta una nota inferior a 2 para la mayoría.*

- labels: Puede especificarse como un valor lógico (TRUE) para indicar si anotaciones van a ser hechas en los "tickmarks" o si un vector de caracteres será colocado en los ticks.

Capítulo 6

Otros Gráficos

6.1 Gráfico de una serie

Adicionalmente a lo visto en el capítulo 4, una serie de tiempo puede ser graficada también como se describe en el siguiente ejemplo:

```
plot(rnorm(10,5,2),axes=F,xlab="")

#Grafica en el plano coordenado 10 puntos tomados aleatoriamente
#de una normal mu=5,
sigma=2,
#pero sin ejes x,y box()
#dibuja una caja alrededor

axis(2,1:10,cex.axis=0.8,las=2)

# dibuja el eje y sobre el lado izquierdo de la
caja,cex.axis comprime en este
#caso el tamaño de las etiquetas de los ticks un 20% y
las=2 ubica las etiquetas
#perpendicular al eje axis(1,1:10,paste(1:10,"Enero",sep="-"),
cex.axis=0.8,las=2)
#Dibuja
el eje x sobre la base de la caja, con 10
#ticks y agrega a cada uno la palabra Enero.
"ces.axis" y "las" como
```

```
# se explic\'o previamente
#una modificaci\'on al anterior:
plot(rnorm(10,5,2),type='l',axes=F,xlab="",ylab="ventas")
box()
axis(2,1:10,cex.axis=0.8,las=2)
axis(1,1:10,paste(1:10,"Enero",sep="-"),cex.axis=0.8,las=2)
```

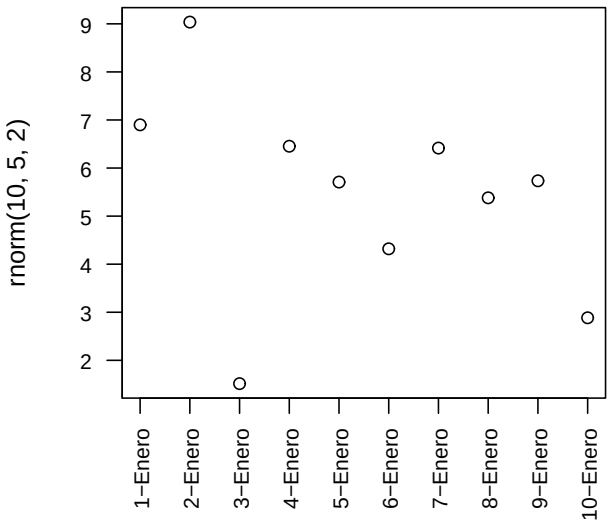

Figura 6.1: *Los datos de una serie simulada para los primeros 10 días de enero.*

6.2 Gráficos de control univariados para el centramiento

Los gráficos de control de calidad univariados para el centramiento, consisten de tres líneas horizontales correspondientes a un nivel medio y los límites

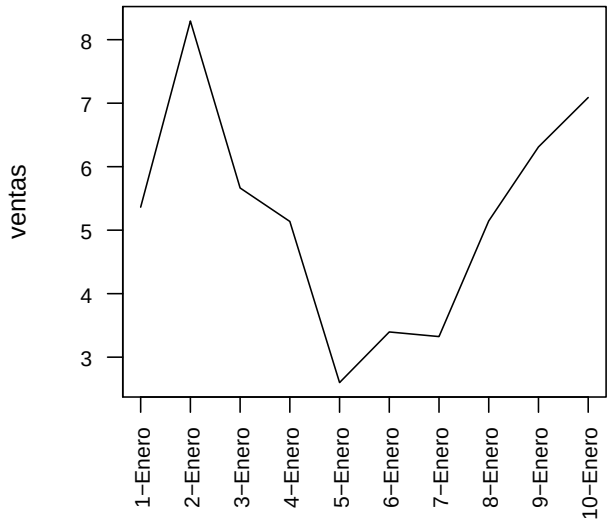

Figura 6.2: *Los mismos datos de la serie simulada para los primeros 10 días de enero, uniendo con líneas los puntos.*

superior e inferior que enmarcan el rango de variabilidad inherente de una característica, sobre este gráfico se ubican los puntos correspondientes al valor de las medias de muestras tomadas en ciertos intervalos de muestreo u observación. Para el caso, asumiendo muestras del mismo tamaño, el gráfico de control $\bar{X}$ para una variable cuya distribución se asume normal con media μ y varianza σ conocidos, una carta con límites 3σ, ser´como sigue (se asume que la matriz de datos se construye con las muestras correspondiendo a las columnas):

```
a<-matrix(rnorm(50,850,16),ncol=10)
b<-matrix(rnorm(50,800,16),ncol=10)
x<-cbind(a,b)
carta.X.bar<-function(x,mu,sigma){
tiempo<-1:ncol(x)
LCL<-mu-3*sigma
UCL<-mu+3*sigma
medias<-c(apply(x,2,mean))

plot(tiempo,medias,type='l',xaxt='n',
xlim=c(0,(length(medias)+2)),ylim=c(min(LCL,min(medias))-2,
max(UCL,max(medias))+2))

axis(1,1:(length(medias)+2),cex.axis=0.8,las=2)
abline(h=mu,lty=3)
text((max(tiempo)+1),mu,paste('mu=',mu),pos=3,font=2,cex=0.8)
abline(h=UCL,lty=3)
text((max(tiempo)+1),UCL,paste('UCL=',UCL),pos=3,font=2,cex=0.8)
abline(h=LCL,lty=3)
text((max(tiempo)+1),LCL,paste('LCL=',LCL),pos=3,font=2,cex=0.8)

for(i in 1:length(medias)){
if(medias[i]>UCL)temp<-4
else if(medias[i]<LCL)temp<-4

else temp<-19

points(tiempo[i],medias[i],pch=temp)
}
```

```
mtext('carta de Medias',side=3,font=2)
}
carta.X.bar(x,850,16)
```

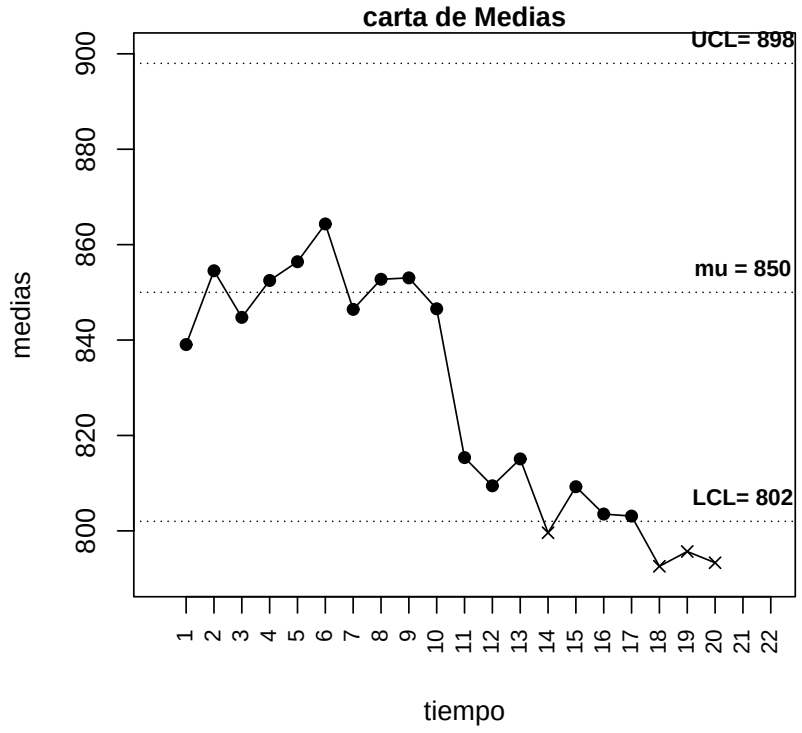

Figura 6.3: *Los puntos por fuera de límites son señalados con una cruz. En este gráfico se observa un cambio brusco del nivel medio de la variable bajo control.*

6.3 Cartas de control multivariado

Entre las varias cartas de control multivariado la más popular es la carta T^2, basada en el estadístico T^2 de Hotelling, para la cual se asume que p

características de calidad correlacionadas son controladas simultáneamente, y que es normal multivariado con vector de medias μ y matriz de varianzas - covarianzas Σ. Asumiendo que estos dos últimos son desconocidos y que se toma una muestra de tamaño n, la carta estaría compuesta por un límite superior, LC, proporcional a un cuantil $Beta_{\frac{p}{2},\frac{n-p-1}{2}}$, mientras que los puntos graficados corresponden a los valores T_i^2 observados en n muestras (las observaciones corresponden a las filas de la matriz de datos):

```
bivariada<-function(n,mu.x,mu.y,sigma.x,sigma.y,ro){
x<-c(rnorm(n,mu.x,sigma.x))
mu.y.x<-mu.y+ro*sigma.y*(x-mu.x)/sigma.x
sigma.y.x<-sigma.y*sqrt(1-ro^2)
y<-c(rnorm(n,mu.y.x,sigma.y.x))
datos<-cbind(x,y)
datos
}
x<-bivariada(30,50,135,16,20,0.8)
carta.T2<-function(x,alpha){
n<-nrow(x)
p<-ncol(x)
LC<-(((n-1)^2)/n)*qbeta((1-alpha),(p/2),(n-p-1)/2)
medias.x<-apply(x,2,mean)
medias<-t(medias.x)
uno<-c(rep(1,nrow(x)))
matriz.uno<-matrix(t(uno))
matriz.medias<-matriz.uno%*%medias
datos.centrados<-x-matriz.medias
matriz.covarianza<-var(x)
inversa<-solve(matriz.covarianza)
aux<-datos.centrados%*%inversa%*%t(datos.centrados)
T2<-diag(aux)
tiempo<-1:length(T2)
plot(tiempo,T2,type='l',xlim=c(0,(length(T2)+1)),
ylim=c(0,max(LC,max(T2))+2),xaxt='n')
axis(1,1:(length(T2)+1),cex.axis=0.8,las=2)
abline(h=LC,lty=3)
```

```
for(i in 1:length(T2)){
temp<-ifelse((T2[i]>LC),4,19)
points(tiempo[i],T2[i],pch=temp)}
text((max(tiempo)-1),LC,paste('LC=',LC),pos=3,font=2,cex=0.8)
mtext(paste('Carta T2','alpha=',alpha),side=3,font=2)
}

carta.T2(x,0.005)
```

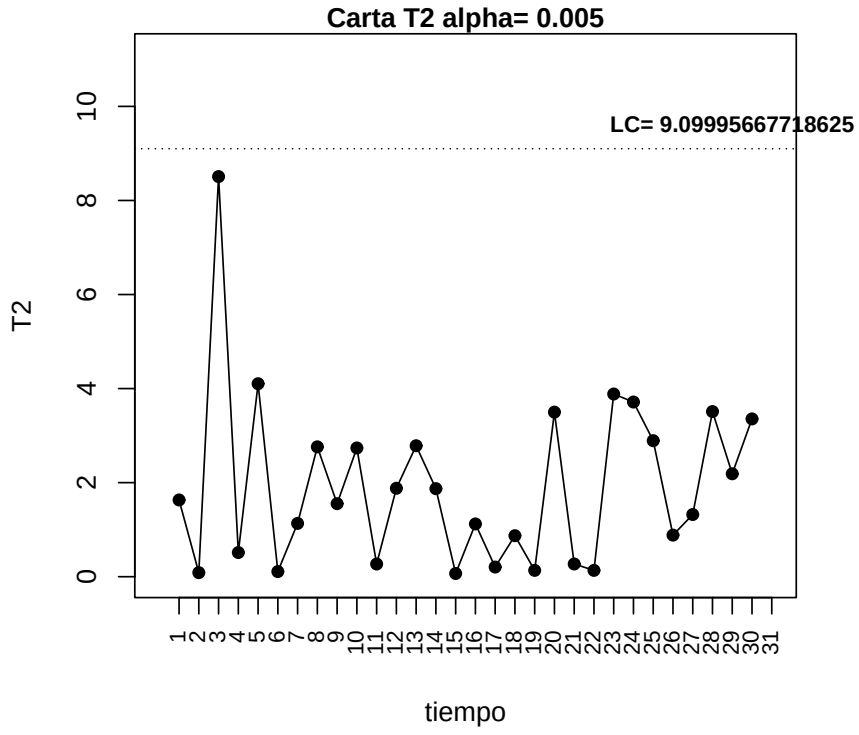

Figura 6.4: *Todos los puntos por debajo del LC indican que el proceso multivariado está bajo control.*

```
#otro ejemplo
```

```
x1<-bivariada(15,50,135,16,20,0.8)
x2<-bivariada(1,100,200,16,20,0.8)
x3<-bivariada(14,50,135,16,20,0.8)
x<-rbind(x1,x2,x3)
x
```

```
          [,1]      [,2]
 [1,]  60.41069 136.07463
 [2,]  52.68355 151.41869
 [3,]  63.16841 146.72261
 [4,]  48.14519 149.99653
 [5,]  64.02661 159.62313
 [6,]  35.10927  87.94832
 [7,]  58.07486 152.85228
 [8,]  44.24636 123.21088
 [9,]  32.01562 118.34291
[10,]  64.83873 154.45834
[11,]  34.06602 104.00438
[12,]  46.07400 126.90240
[13,]  27.15364 128.06738
[14,]  41.30233 125.80879
[15,]  43.74436 125.03584
[16,] 104.18832 204.93275
[17,]  41.82251 123.88981
[18,]  47.61095 145.69089
[19,]  59.46790 123.49907
[20,]  24.23434 110.48738
[21,]  56.71484 128.98529
[22,]  44.33605 139.91945
[23,]  62.57244 149.16832
[24,]  51.45150 137.84961
[25,]  63.37475 136.73632
[26,]  35.28002 111.21907
[27,]  69.77807 175.80549
[28,]  23.23128  95.17229
[29,]  55.85678 154.86840
[30,]  47.63253 113.70498
```

```
carta.T2(x,0.005)
```

6.4 Graficando una elipse

La siguiente función permite graficar una elipse dado su centro, y una matriz A positiva definida de dimensión 2x2,cuya inversa permite definir la forma cuadrática asociada a la elipse que se desea graficar (la fila i contiene las coordenadas (x, y) del punto i). Como resultado esta función produce una matriz X de dimensión kx2 cuyas filas contienen las coordenadas de los k

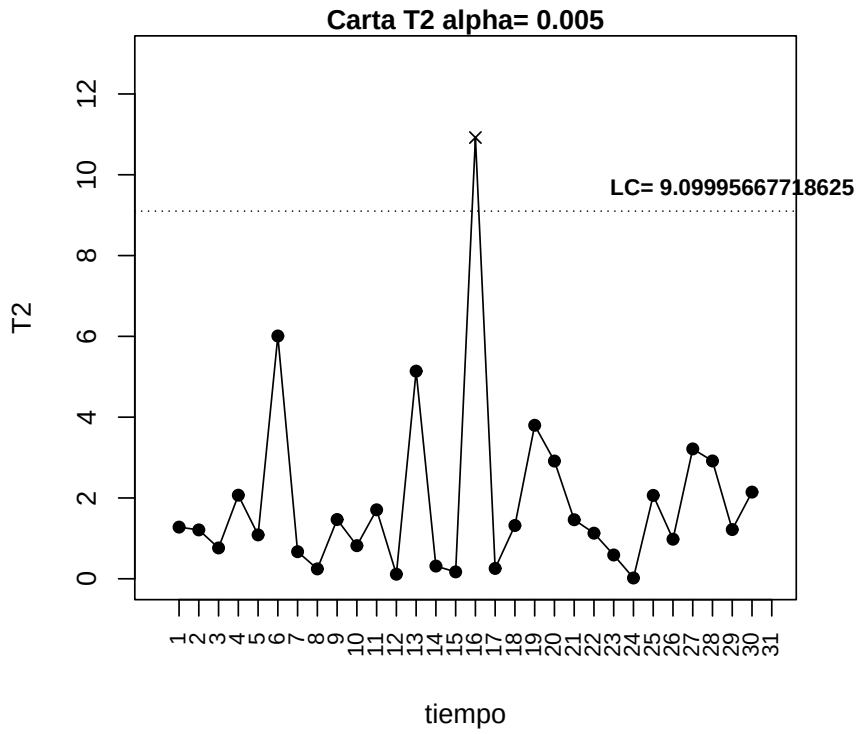

Figura 6.5: *La observación 16 arriba del LC indican que el proceso multivariado estuvo fuera control en dicho periódo.*

puntos sobre la elipse $(\mathbf{y}-\mathbf{m})^{\mathbf{t}}\mathbf{A}^{-\mathbf{1}}(\mathbf{y}-\mathbf{m}) = c^2$. Recuerde que el coseno entre dos vectores define la correlación entre las variables asociadas a dichos vectores

```
elipse<- function(A, m, const, k){

r <- A[1, 2]/sqrt(A[1, 1] * A[2, 2])
Q <- matrix(0, 2, 2) # construye una matriz nula Q

Q[1, 1] <- sqrt(A[1, 1] %*% (1+r)/2)  # transformacion del circulo
Q[1, 2] <- -sqrt(A[1, 1] %*% (1-r)/2) # unitario a una elipse
Q[2, 1] <- sqrt(A[2, 2] %*% (1+r)/2)
Q[2, 2] <- sqrt(A[2, 2] %*% (1-r)/2)

alpha <- seq(0, by = (2 * pi)/k, length = k) # define angulos
                                             para graficar

Z <- cbind(cos(alpha), sin(alpha)) # Define coordenadas
                                   #de puntos sobre c\'irculo
                                   #unitario

X <- t(m + const * Q %*% t(Z)) # Define coordenadas de puntos
                               #sobre la elipse
X
}
```

Los argumentos de esta función son los siguientes:

- A: Una matriz positiva definida simétrica de dimensión 2x2 (la diagonal debe ser de valores positivos)
- m: Un vector columna de longitud 2 que define el centro de la elipse.
- const: Una constante positiva (el valor de c de la forma cuadrática
- k: Número de puntos sobre la elipse (como mínimo debe ser 2)

Para llamar la función y graficar se procede de la siguiente manera:

```
X <- elipse(A, m, const, k) #Asignamos al objeto X
                            #los puntos de elipse calculados
                            con la funcion win.graph()
                            #para abrir una ventana grafica

plot(X[,1], X[,2],type='l',xlab=) #grafica la elipse
```

Ejemplo 1: Sea la elipse $2x^2 + \sqrt{3}xy + y^2 = 8$, con centro en el origen y la cual tiene asociada la matriz:

$$\mathbf{B} = \begin{pmatrix} 2 & \sqrt{3}/2 \\ \sqrt{3}/2 & 1 \end{pmatrix}$$

tal que

$$\mathbf{X^t B X} = 2x^2 + \sqrt{3}xy + y^2$$

con

$$\mathbf{X} = [\ x \quad y\ ]^t$$

entonces para graficarla se procede de la siguiente manera:

```
B<-matrix(c(2,sqrt(3)/2,sqrt(3)/2,1),ncol=2)
A<-solve(B)
m<-c(0,0)
const<-sqrt(8)
#C^{2}corresponde a 8, entonces C=8
k<-1000
X <- elipse(A, m, const, k)
win.graph()
plot(X[,1], X[,2],type='l',xlab='x',ylab='y',
main=expression(paste("elipse",sep="
",2*x^2+sqrt(3)*x*y+y^2==8)))
```

Ejemplo 2: Sea la elipse $41x^2 - 84xy + 76y^2 = 169$, con centro en el origen y la cual tiene asociada la matriz:

$$\mathbf{B} = \begin{pmatrix} 41 & -42 \\ -42 & 76 \end{pmatrix}$$

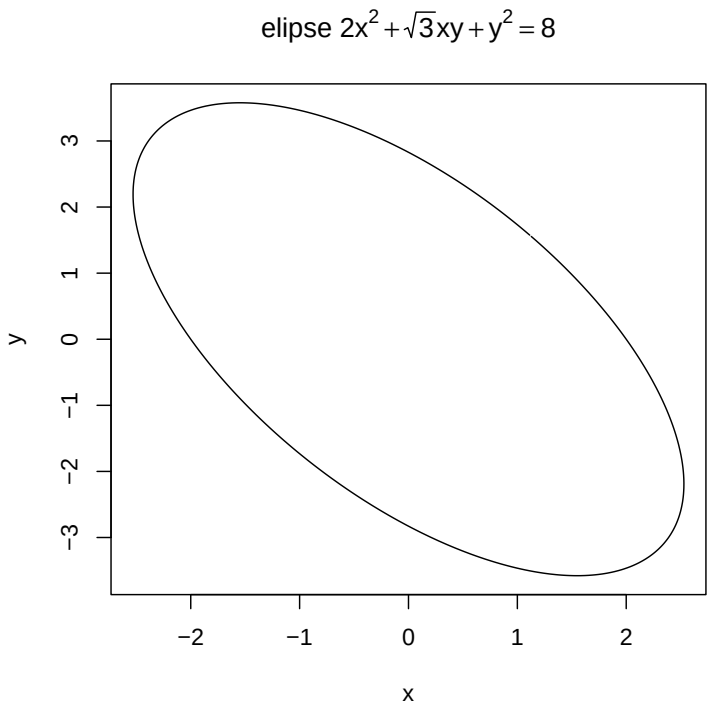

Figura 6.6:

tal que

$$\mathbf{X^t B X} = 2x^2 + \sqrt{3}xy + y^2$$

con

$$\mathbf{X} = [\ x \quad y\ ]^t$$

entonces

```
B<-matrix(c(41,-42,-42,76),ncol=2)
A<-solve(B)
m<-c(0,0)
const<-13 #C^{2} corresponde a
          #169, entonces C=13

k<-1000
X <- elipse(A, m, const, k)

win.graph()
plot(X[,1], X[,2],type='l',xlab='x',ylab='y',
 main=expression(paste("elipse",sep="
",41*x^2-84*x*y+76*y^2==169)))
```

6.5 Graficando elipses de confianza del $(1-\alpha)100\%$ para un conjunto de n observaciones de una distribución normal bivariada

El siguiente procedimiento, es una función que grafica elipses de confianza con base en el estadí stico F, desarrollado por Roger Koenker, (Department of Economics, University of Illinois) denominada *confelli*, la cual grafica una elipse con matriz de covarianza C, con centro en b (vector de medias), y que por defecto contiene el 95% de las observaciones con base en la distribución F(2,df), donde df=n-2.

Nota: Las dos últimas líneas de la función son modificaciones al programa original:

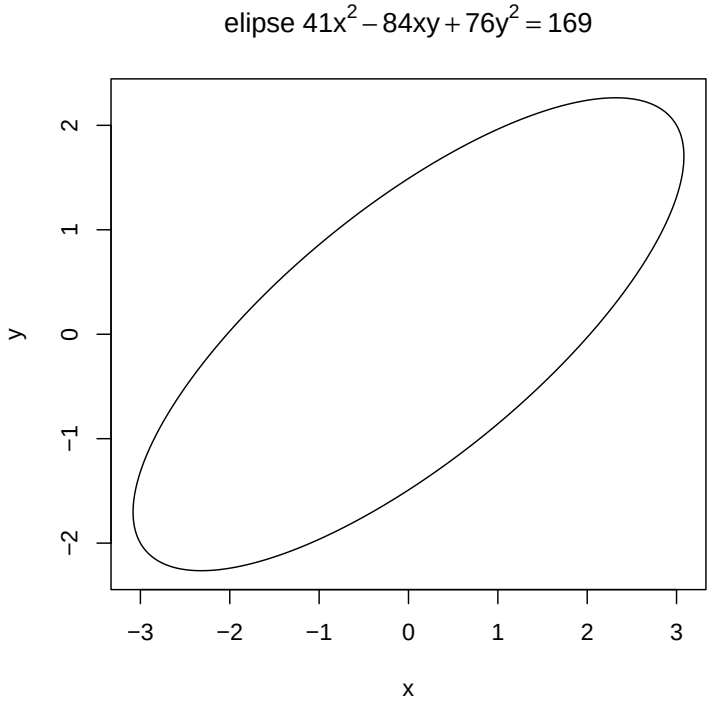

Figura 6.7:

```
# Sent to S-NEWS on May 19, 1999 by Roger Koenker

# Department of Economics

# University of Illinois

# Champaign, IL 61820
# url: http://www.econ.uiuc.edu
# email roger@ysidro.econ.uiuc.edu
# vox: 217-333-4558
# fax: 217-244-6678.

confelli <- function(b, C, df, level = 0.95, xlab = "",
ylab = "", add=T, prec=51)

{
    d <- sqrt(diag(C))
    dfvec <- c(2, df)
    phase <- acos(C[1, 2]/(d[1] * d[2]))
    angles <- seq( - (pi), pi, len = prec)
    mult <- sqrt(dfvec[1] * qf(level, dfvec[1], dfvec[2]))
    xpts <- b[1] + d[1] * mult * cos(angles)
    ypts <- b[2] + d[2] * mult * cos(angles + phase)
    if(add) lines(xpts, ypts)
    else plot(xpts, ypts, type = "l", xlab = xlab, ylab = ylab)
    a<-round(runif(1,1,51))
    text(xpts[a], ypts[a],paste(level),adj=c(0.5,0.5),font=2,cex=0.7)
}
```

Ejemplo 1: Se generan 50 observaciones de una normal bivariada con matrix de covarianza

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

y vector de medias

$$\mu = [\ 0 \quad 0\ ]^t$$

```
C<-matrix(c(1,0,0,1),ncol=2)
b<-c(0,0)
x<-rnorm(50)
y<-rnorm(50)
plot(x,y,xlim=c(-4,4),ylim=c(-4,4),cex=0.5)
confelli(b,C,df=48)
confelli(b,C,df=48,level=0.8)
confelli(b,C,df=48,level=0.5)
title(main=expression(paste('Elipses de confianza para una
normal bivariada',sep="
",Sigma==I,sep=" ","",sep=" ",mu=='c(0,0)')),cex.main=0.9)
```

Ejemplo 2: Se generan 50 observaciones de una normal bivariada con matrix de covarianza

$$\mathbf{\Sigma} = \begin{pmatrix} 1 & 0.7 \\ 0.7 & 1 \end{pmatrix}$$

y vector de medias

$$\mu = [\ 1 \quad 2\ ]^t$$

```
C<-matrix(c(1,0.7,0.7,1),ncol=2)
b<-c(1,2)
bivariada<-function(n,mu.x,mu.y,sigma.x,sigma.y,ro){
x<-c(rnorm(n,mu.x,sigma.x))
mu.y.x<-mu.y+ro*sigma.y*(x-mu.x)/sigma.x
sigma.y.x<-sigma.y*sqrt(1-ro^2)
y<-c(rnorm(n,mu.y.x,sigma.y.x))
datos<-cbind(x,y)
datos
}
datos<-bivariada(50,b[1],b[2],1,1,0.7)
x<-datos[,1]
y<-datos[,2]
plot(x,y,xlim=c(-2,4),ylim=c(-1,5),cex=0.5)
confelli(b,C,df=48)
confelli(b,C,df=48,level=0.8)
confelli(b,C,df=48,level=0.5)
```

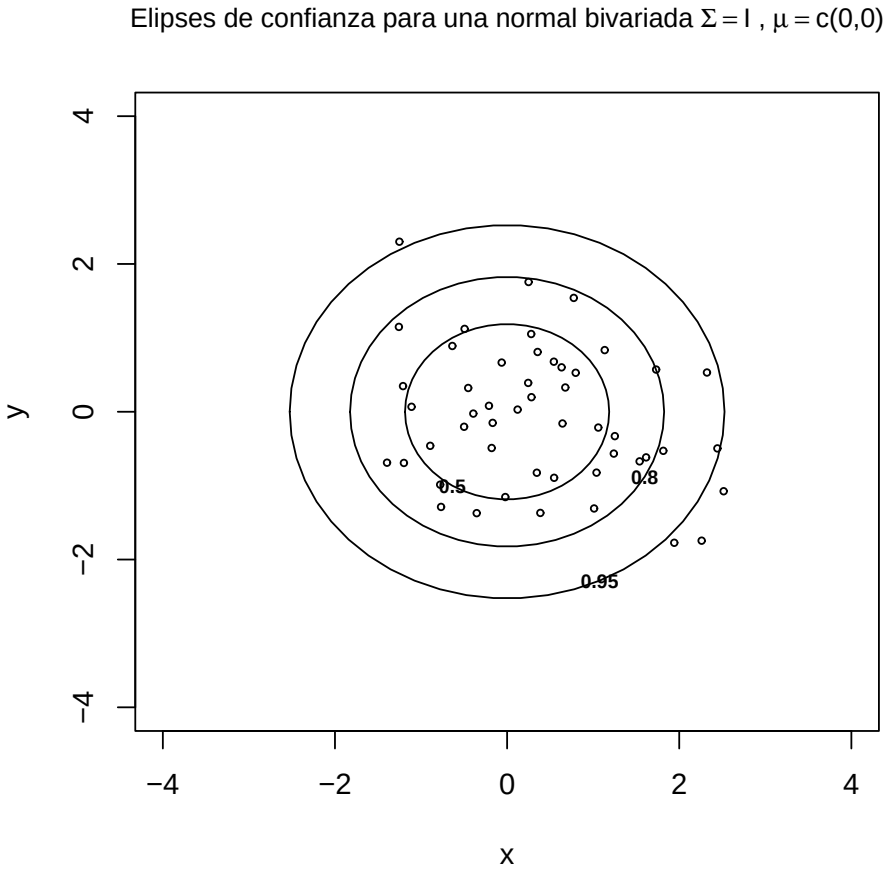
Elipses de confianza para una normal bivariada Σ = I , μ = c(0,0)
y
x
4
2
0
−2
−4
−4
−2
0
2
4
0.5
0.8
0.95

Figura 6.8:

```
title(main='Elipses de
confianza para una normal bivariada',cex.main=0.9)
par(oma=c(1,1,3.5,1),new=T,font=2,cex=0.9)
mtext(outer=T,expression(paste(ro==0.7,sep="
","",sep=" ",sigma(x)==sigma(y)==1,sep=" ",mu=='c(1,2)')), side=3)
```

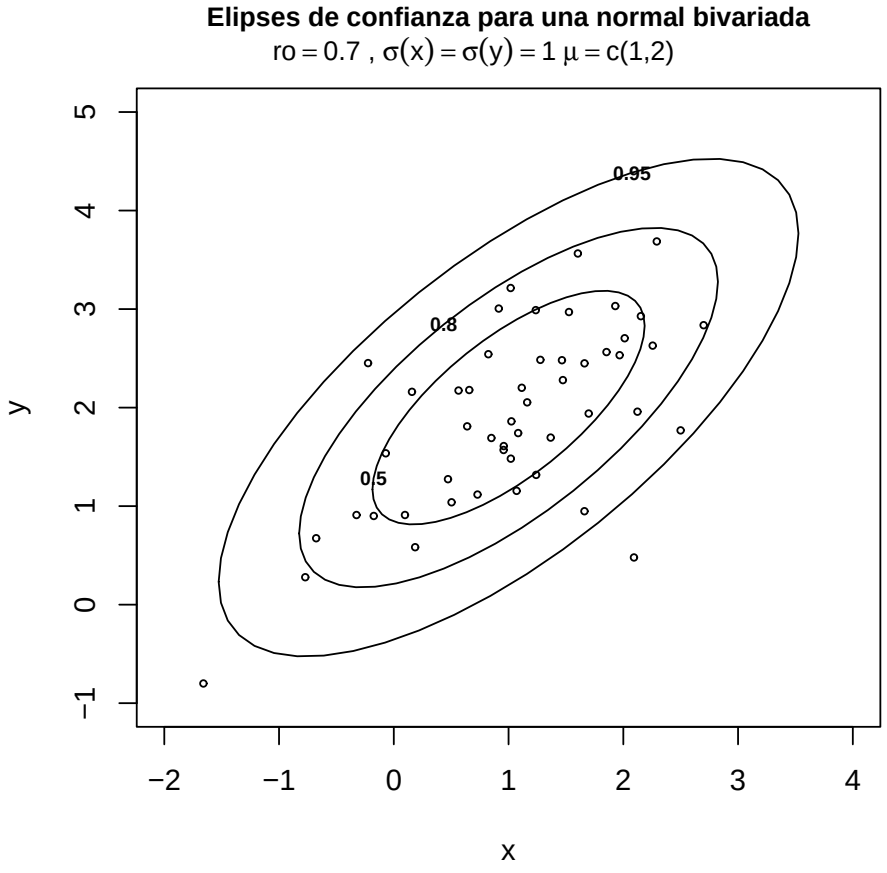

Figura 6.9:

6.6 Opciones especiales

6.6.1 Guardar y llamar gráficos

Con recordPlot() se puede guardar un gráfico en *R* y llamarlo cuando se desee, por ejemplo:

```
plot(1:10)
abline(h=5)
abline(v=5)
t.plot<-recordPlot()
t.plot
```

6.6.2 función `layout` (especificación de arreglos gráficos complejos)

Esta función divide el dispositivo (ventana graficadora) en tantas filas y columnas haya en la matriz especificada en el argumento 'mat', con los anchos de columna y los altos de fila especificados en los respectivos argumentos;la i-ésima figura es ubicada en una región compuesta de un subconjunto de estas filas y columnas, con base en las filas y columnas en las cuales i ocurre en 'mat':

```
layout(mat, widths = rep(1, dim(mat)[2]),
heights= rep(1, dim(mat)[1]),respect= FALSE)

layout.show(n = 1)

lcm(x)
```

Valor: 'layout' retorna el número de figuras, N, a presentar en el dispositivo.

Argumentos:

- mat: Es una matriz con la cual se especifica la ubicación de las siguientes N figuras en la salida del dispositivo. Cada valor en la matriz debe ser un entero positivo o cero. Si el número de figuras N es el entero positivo más grande en la matriz, entonces los enteros $\{1, ..., N-1\}$ deben aparecer por lo menos una vez en la matriz.

- widths: Es un vector de valores para los anchos de columnas en el dispositivo. Los anchos relativos son especificados con valores numéricos. Los anchos absolutos (en centímetros) son especificados con la función **lcm()** -

- heights: Es un vector de valores para los altos de filas en el dispositivo. Los altos relativos y absolutos pueden ser especificados como se indicó en 'widths'.

- respect: Un valor lógico o una matriz. En el último caso debe tener las mismas dimensiones que 'mat' y cada valor en la matriz debe ser 0 ó 1. Este argumento controla si una unidad de ancho de columna es la misma medida física en el dispositivo de la unidad de alto de fila.

- n: El número de figuras a graficar.

- x: Una dimensión a ser interpretada como un número de centímetros.

- 'layout.show(n)': Esta función grafica parte o el layout actual, esto es los perfiles de las siguientes n figuras

- 'lcm': Es una funión trivial, usada como la interface para especificar dimensiones absolutas para los argumentos 'widths' y 'heights' de 'layout(...)'.

Autor: Paul R. Murrell

Referencias:

- Murrell, P. R. (1999) Layouts: A mechanism for arranging plots on a page. Journal of Computational and Graphical Statistics, 8, 121-134. Chapter 5 of Paul Murrell's Ph.D. thesis.

- R Documentation

Ver también: 'par(mfrow=..)', 'par(mfcol=..)' y 'par(mfg=..)'.

Ejemplo 1: Dividir el dispositivo en dos filas y dos columnas, ubicar la figura 1 en toda la fila 1, ubicar la figura 2 en la interseccón de la columna 2 y la fila 2:

```
#guardar los valores por defecto, para restaurarlos luego....

def.par <- par(no.readonly = TRUE)


a<-matrix(c(1,1,0,2), 2, 2, byrow = TRUE)
> a
     [,1] [,2]
[1,]    1    1
[2,]    0    2

layout(a)

#para mostrar todas las regiones del
#anterior layout, se usa n=2:

layout.show(2)

#resetear parametros graficos
#a valores por defecto:
par(def.par)
```

Ejemplo 2: Una aplicación del anterior diseño:

```
layout(a)
plot(rexp(30))
title(main='Un ejemplo de la presentación
de las figuras\ncon
layout(matrix(c(1,1,0,2), 2, 2, byrow = TRUE))',cex.main=0.8)
plot(rnorm(30))
```

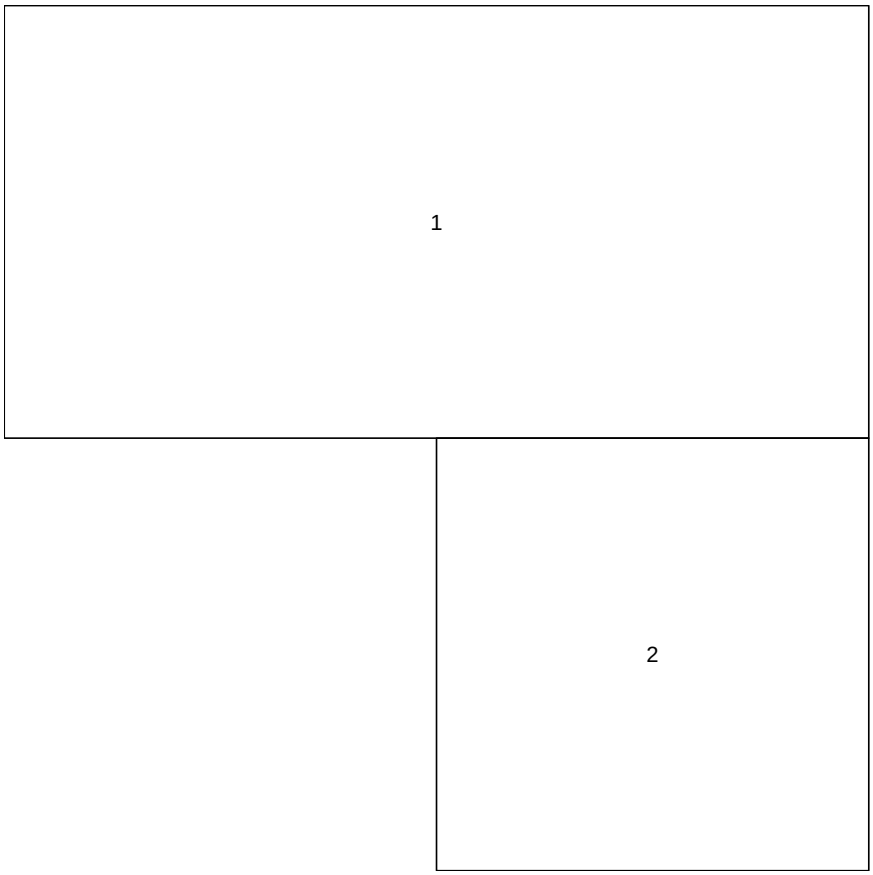

Figura 6.10: *División del dispositivo logrado con layout(matrix(c(1,1,0,2), 2, 2, byrow = TRUE)*

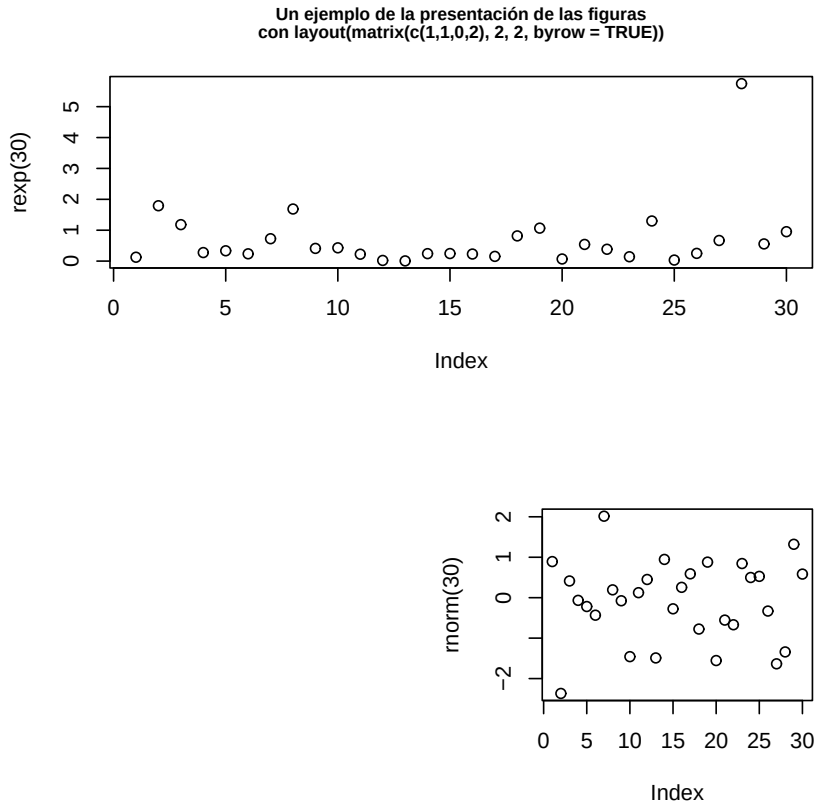

Figura 6.11:

Ejemplo 3: Dividir el dispositivo en dos filas y dos columnas; ubicar la figura 1 y la figura 2 como en el ejemplo anterior, pero respetar relaciones entre anchos y altos:

```
nf <- layout(matrix(c(1,1,0,2), 2, 2, byrow=TRUE), respect=TRUE)
layout.show(nf)
```

Ejemplo 4: Crear un sola figura que sea un cuadrado de 5cm:

```
nf <- layout(matrix(1), widths=lcm(5), heights=lcm(5))
layout.show(nf)
```

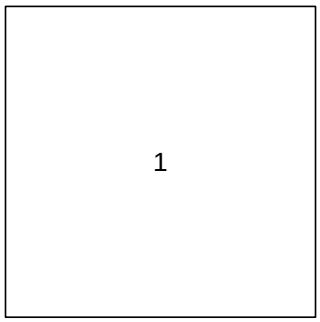

Figura 6.12: *Efecto de layout(matrix(1), widths=lcm(5), heights=lcm(5))*

Ejemplo 5: Con la función layout es posible obtener un diseño en el cual tres gráficos son ordenados de modo que el superior quede sobre otros dos; para esto se hace lo siguiente:

```
nf<-layout(matrix(c(1,1,2,3), 2, 2, byrow=TRUE),respect=TRUE)
```

```
plot(1:10)
plot(rnorm(10,0,1))
plot(rlnorm(10,0,1))
```

Lo anterior produce el siguiente gráfico:

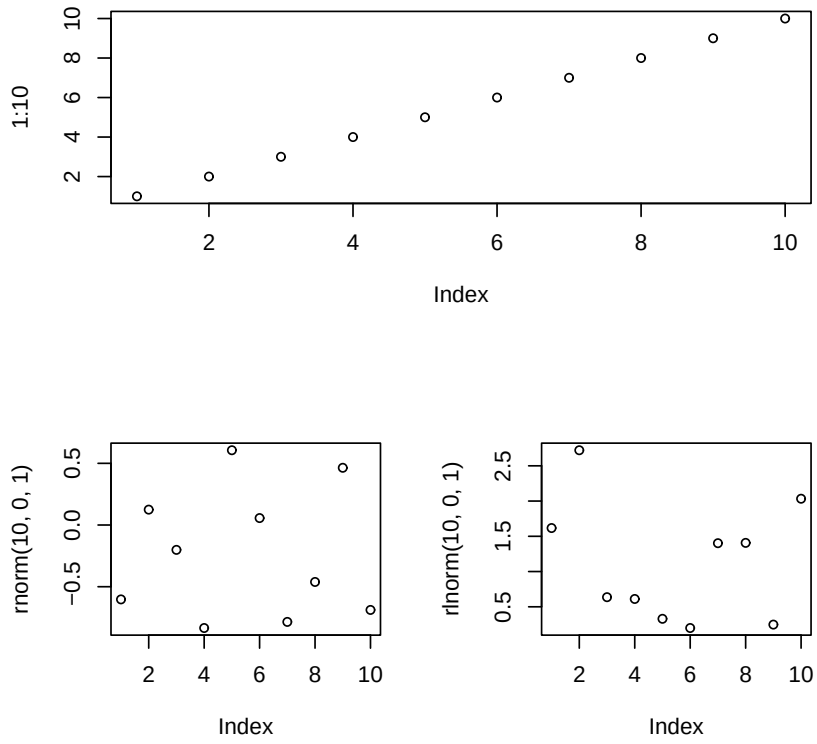

Figura 6.13: *Efecto de layout(matrix(c(1,1,2,3), 2, 2, by-row=TRUE),respect=TRUE)*

Ejemplo 6: Es posible obtener los tres gráficos anteriores con el mismo ordenamiento pero todos de aproximadamente el mismo ancho, de la siguiente manera:

```
layout(matrix(c(1,1,2,3), 2, 2, byrow=TRUE),respect=TRUE)
par(mar=c(5.1, 12.1, 1.4, 10.1))
plot(rnorm(10))
par(mar=c(5.1, 4.1, 1.4, 2.1))
plot(rnorm(10))
plot(rnorm(10))
```

Lo cual produce el siguiente gráfico:

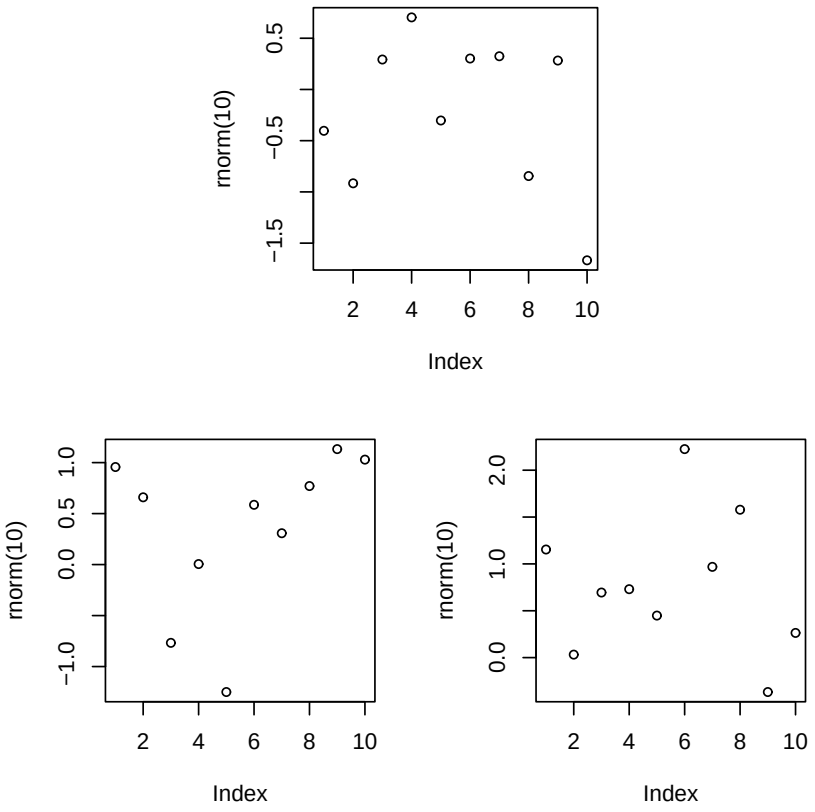

Figura 6.14: *Efecto de layout(matrix(c(1,1,2,3), 2, 2, byrow=TRUE),respect=TRUE) con par(mar)*

Ejemplo 7: Lo anterior también puede lograrse de la siguiente forma:

```
nf <- layout(rbind(c(0,1,1,0), c(2,2,3,3)))
plot(1:10)
plot(rnorm(10,0,1))
plot(rlnorm(10,0,1))
```

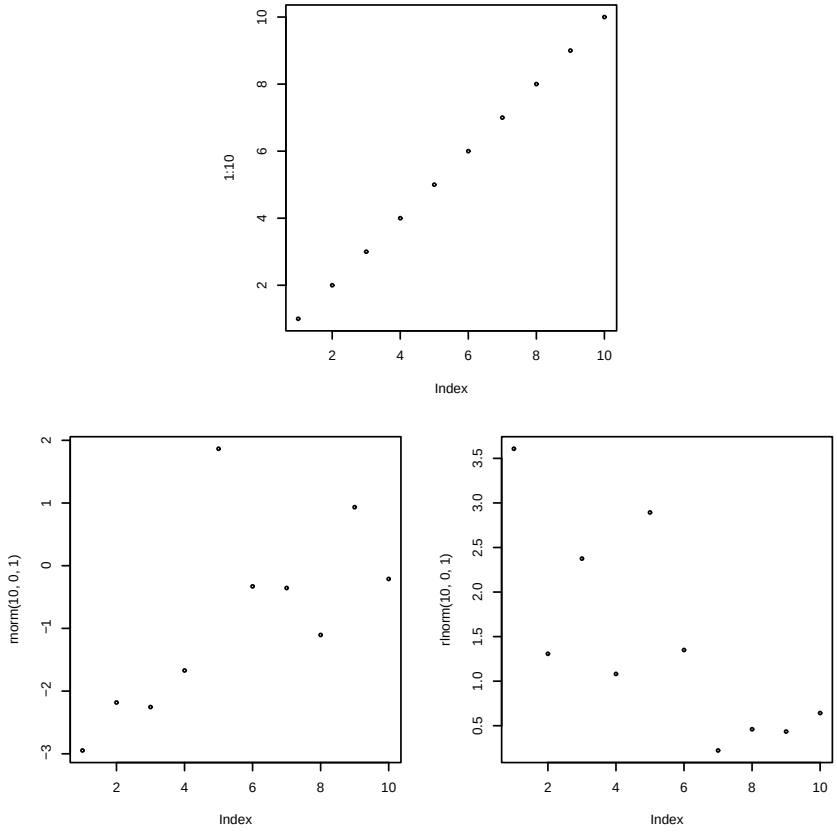

Figura 6.15: *Efecto de layout(rbind(c(0,1,1,0), c(2,2,3,3)))*

Ejemplo 8: Otra presentación, en la cual dos gráficos son presentados en columna, y centrados en la ventana graficadora es posible de la forma siguiente:

```
nf <- layout(rbind(c(0,1,1,0), c(0,2,2,0)))
```

```
plot(rnorm(10))
plot(rlnorm(10))
```

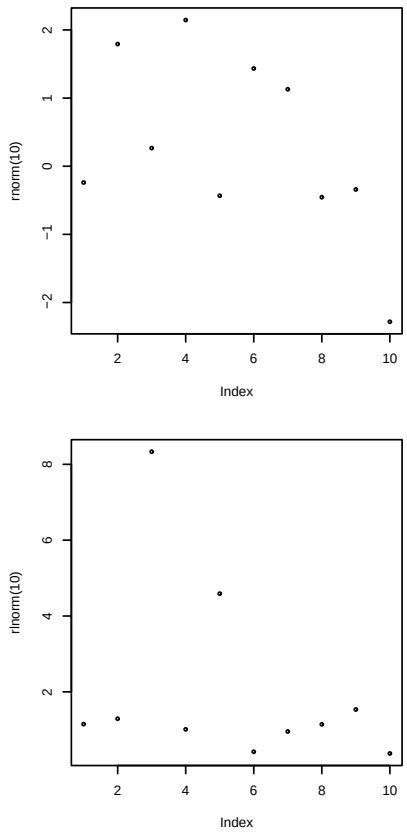

Figura 6.16: *Efecto de layout(rbind(c(0,1,1,0), c(0,2,2,0)))*

Ejemplo 9: Crear un diagrama de dispersion con histogramas marginales:

```
x <- pmin(3, pmax(-3, rnorm(50)))
y <- pmin(3, pmax(-3, rnorm(50)))

xhist <- hist(x, breaks=seq(-3,3,0.5), plot=FALSE)
yhist <- hist(y, breaks=seq(-3,3,0.5),
```

```
plot=FALSE)

top <- max(c(xhist$counts, yhist$counts))
xrange <- c(-3,3)
yrange <- c(-3,3)

b<-matrix(c(2,0,1,3),2,2,byrow=TRUE)
b
     [,1] [,2]
[1,]    2    0
[2,]    1    3

#crear la division deseada para el dispositivo:

nf <- layout(b, widths=c(3,1), heights=c(1,3), respect=TRUE)

#notar que los valores de la matriz b indican
#la posicion de las figuras sucesivas

#Definir lineas de margenes a los
#lados de la primera figura:

par(mar=c(4,4,1,1))

#Dibujar el scatterplot:

plot(x, y, xlim=xrange, ylim=yrange, xlab="x", ylab="y")

#Definir lineas de margenes segunda figura:

par(mar=c(0,4,1,1))

#dibujar el histograma marginal en x:

barplot(xhist$counts,axes=FALSE, ylim=c(0, top), space=0)

#Como esta es la figura mas hacia arriba,
#El titulo de todo el grafico
```

```
#puede invocarse
a continuacion:

title(main='Scatterplot con histogramas marginales',font=2)

#Definir lineas de margenes tercera figura:

par(mar=c(4,0,1,1))

#dibujar el histograma marginal en y:
barplot(yhist$counts, axes=FALSE, xlim=c(0, top),
space=0, horiz=TRUE)
```

Ejemplo 10: Crear un diagrama de dispersion con boxplots marginales:

```
c<-matrix(c(0,2,3,1),2,2,byrow=TRUE)
nf <- layout(c, widths=c(0.8,3), heights=c(0.8,3),
respect=TRUE)

par(mar=c(4,1,1,4))
plot(x, y, xlim=xrange, ylim=yrange, yaxt='n',xlab="x",
 ylab="")
axis(4,-3:3,as.character(seq(-3,3,1)))

par(mar=c(0,1,1.5,4))
boxplot(x,horizontal=TRUE,xaxt='n',col="gray")
title(main='Scatterplot con boxplot marginales',font=2)

par(mar=c(4,1,1,0))
boxplot(y,yaxt='n',col="gray")
```

ejemplo 11: Crear un diagrama de dispersion con histogramas y boxplots marginales:

```
d<-matrix(c(3,0,0,2,0,0,1,4,5),ncol=3,nrow=3,byrow=T)
```

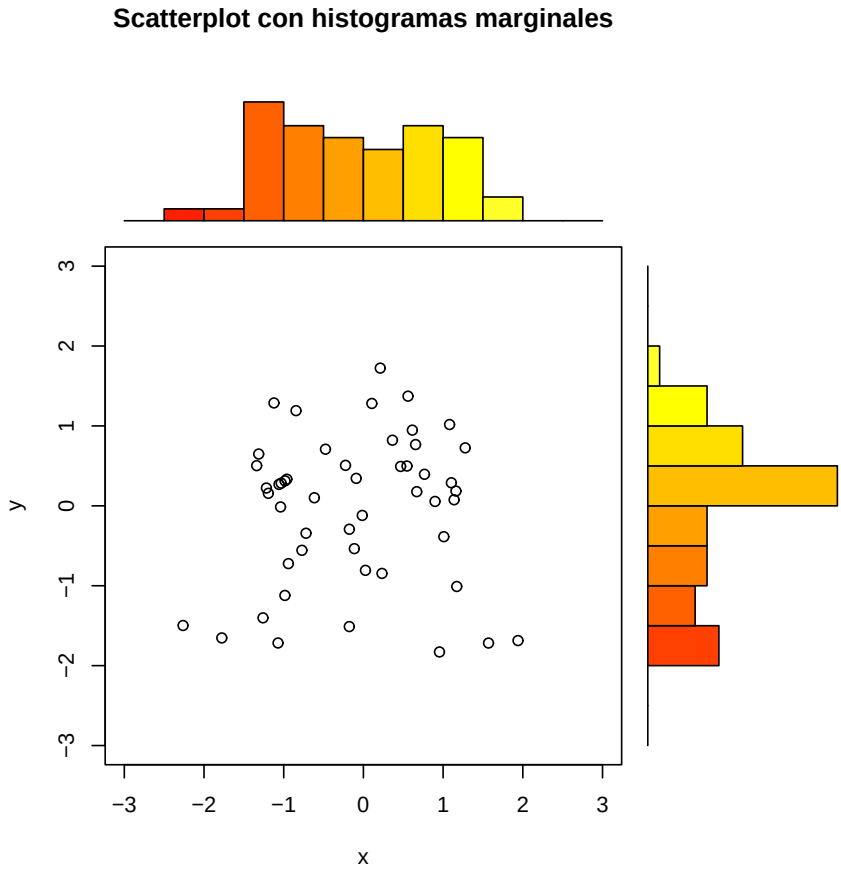

Figura 6.17: *Un gráfico como este permite visualizar simultáneamente la distribución univariada como las posibles asociaciones entre dos variables*

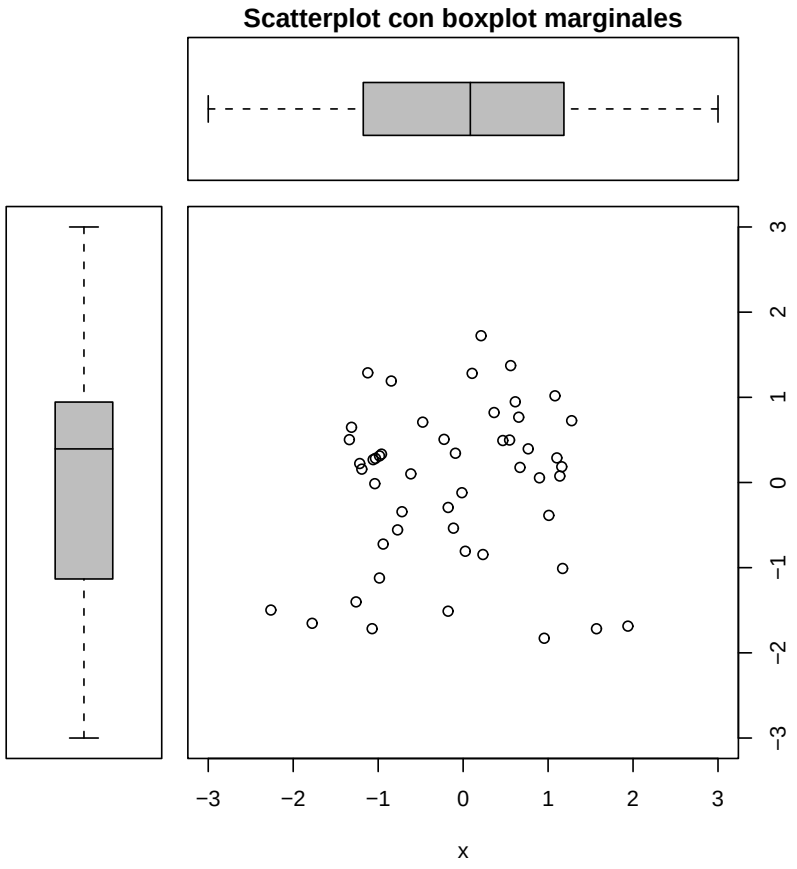

Figura 6.18: *Como en el gráfico previo, la información univariada y bivariada puede ser analizada simultáneamente, en cuanto a la simetría y la dispersión de cada variable y la distribución conjunta.*

```
d
      [,1] [,2] [,3]
[1,]    3    0    0
[2,]    2    0    0
[3,]    1    4    5

nf<-layout(d,widths=c(3,0.5,1.3),heights=c(1.3,0.5,3),
respect=T)

x <- pmin(3, pmax(-3, rnorm(50)))
y <- pmin(3, pmax(-3, rnorm(50)))

xhist <- hist(x, breaks=seq(-3,3,0.5), plot=FALSE)
yhist <- hist(y, breaks=seq(-3,3,0.5),
plot=FALSE)

top <- max(c(xhist$counts, yhist$counts))
xrange <- c(-3,3)
yrange <- c(-3,3)

par(mar=c(4,4,1,1))
plot(x, y, xlim=xrange, ylim=yrange,xlab="x", ylab="y")

par(mar=c(0,4,0.5,1))
boxplot(x,horizontal=TRUE,xaxt='n',col="gray")

par(mar=c(0,4,1,1))
barplot(xhist$counts,axes=FALSE, ylim=c(0, top), space=0)

title(main='SCATTERPLOT CON HISTOGRAMAS Y
BOXPLOTS MARGINALES',cex.main=0.9,font=2)

par(mar=c(4,0,1,0.5))
boxplot(y,yaxt='n',col="gray")
par(mar=c(4,0,1,1))
barplot(yhist$counts, axes=FALSE,
xlim=c(0, top), space=0, horiz=TRUE)
```

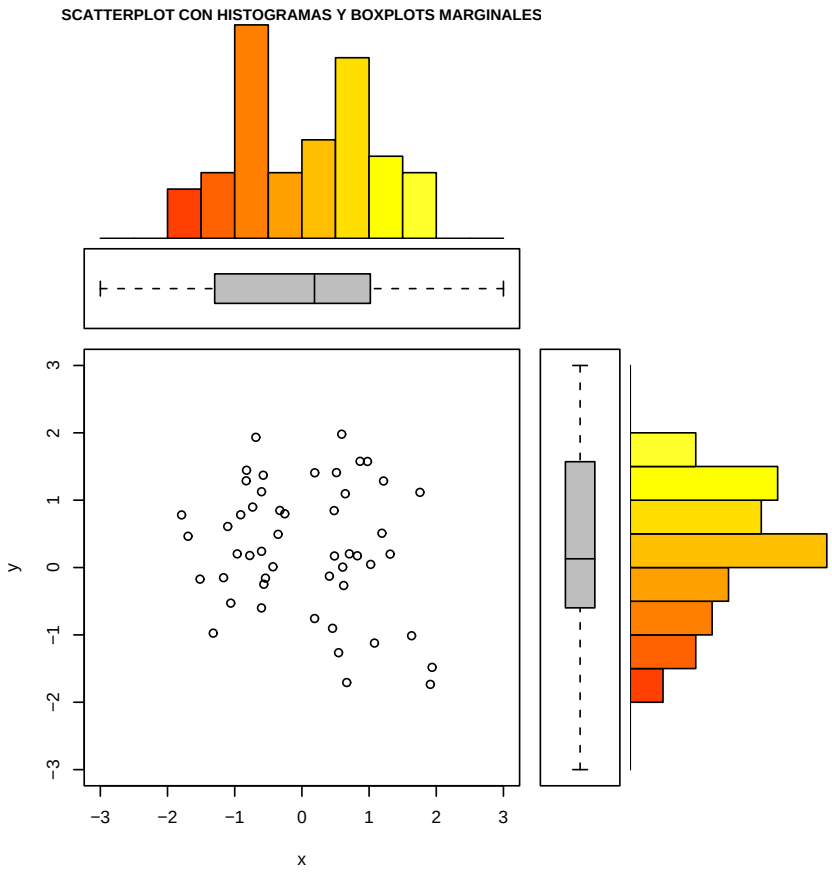

Figura 6.19: *Un gráfico más completo que los dos previos resulta bastante informativo, al permitir visualizar simultáneamente la distribución, la simetría o asimetría, la dispersión univariada y la asociación y comportamiento multivariado, incluso señalar posibles outliers.*

6.6.3 Superposición de curvas a un histograma

Para graficar una curva sobrepuesta a un histograma se puede proceder como sigue:

```
hist(rnorm(100),freq=F)
curve(dnorm(x),add=T)
```

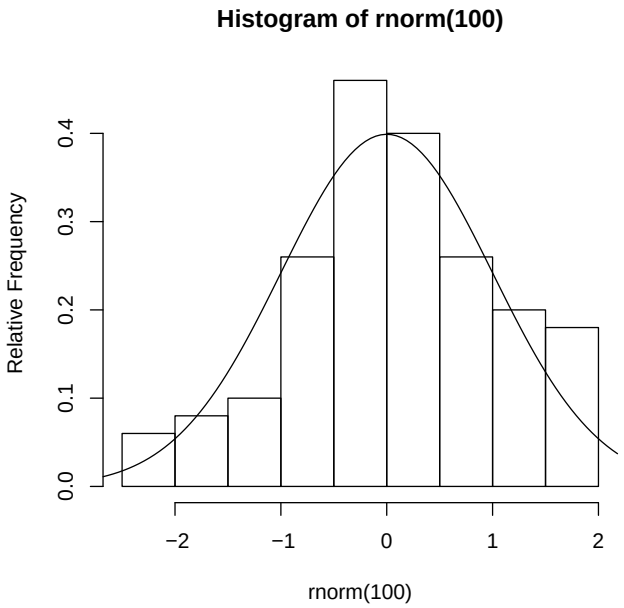

Figura 6.20: *Note que es necesario que las frecuencias sean las relativas y no las absolutas.*

6.6.4 Efectos de sombreado en gráficos tridimensionales

A continuación se presenta un ejemplo en el cual a una superficie tridimensional se le aplica efecto de sombreado:

```
# Creación de la superficie f(x,y) = x^2 - y^2
nx <- 21
ny <- 21

x <- seq(-1, 1, length = nx)
y <- seq(-1, 1, length = ny)

z <- outer(x, y, function(x,y) x^2 - y^2)

#  Promediar los valores en cada esquina
#y escalar a un valor en [0, 1].  Esto se usará
#  para seleccionar un gris para colorear la superficie.

hgt <- 0.25 * (z[-nx,-ny] + z[-1,-ny] + z[-nx,-1] + z[-1,-1])
hgt <- (hgt - min(hgt))/ (max(hgt) - min(hgt))

#Graficación de la superficie con los
#efectos de color especificados.

persp(x, y, z, col = gray(1 - hgt), theta = 35)
persp(x, y, z, col =
cm.colors(10)[floor(9* hgt + 1)], theta = 35)
```

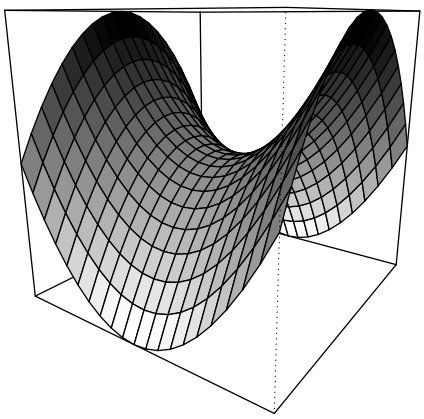

Figura 6.21: *Efectos de sombreado en tonos de gris en un gráfico tridimensional.*

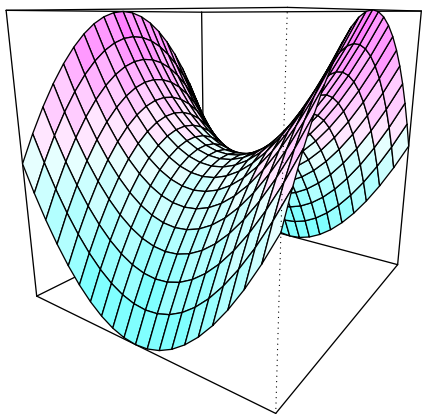

Figura 6.22: *Efectos de sombreado en tonos azules en un gráfico tridimensional.*

Capítulo 7

Apéndices

7.1 Parámetros gráficos (función par)

La función "par", puede ser invocada previamente a cualquier ventana graficadora o gráfico en particular. Igualmente, los argumentos de dicha función, pueden (si son permitidos de acuerdo al tipo de gráfico) ser especificados en un gráfico dado. Los parámetros gráficos generalmente se especifican en la forma de "etiqueta=valor" ($< tag >= value$), donde etiqueta corresponde al nombre del argumento, como se verá a continuación:

```
par(..., no.readonly = FALSE)

<highlevel plot> (..., <tag> = <value>)
```

Los argumentos son:

- no.readonly: Argumento lógico; if es TRUE y no hay otros argumentos, sólo son retornados (listados) los parámetros gráficos que puedan ser ajustados por una llamada subsecuente "par(.)"
- adj: El valor que tome determina la forma en la cual las cadenas (strings) de texto son justificadas.
    1. 0 para justificar el texto a la izquierda,

2. 0.5 para centrar el texto, y
3. 1 para justificar el texto a la derecha.

también es permitido cualquier valor entre $[0, 1]$ y en muchos dispositivos también funcionan valores por fuera de este intervalo. El valor de este argumento para la funión "text" debe darse en la forma $adj = c(x, y)$ para determinar la justificación del texto tanto en la dirección x e y.

- ann: Argumento lógico. Si se ajusta a FALSE, entonces en las funciones de graficación de alto nivel no se realiza anotación sobre los ejes, sólo producen los ejes con sus títulos y el respectivo gráfico
- ask: Argumento lógico. Si es TRUE, el usuario es preguntado por el input antes que una nueva figura sea dibujada.
- bg: Para especificar el color a ser usado en el fondo de los gráficos. Ver más adelante la especificación de los colores.
- bty: Una cadena de caracteres que determina el tipo de caja que será dibujada alrededor de los gráficos. Si se especifica como uno de los siguientes caracteres (entre comillas):"o", "l", "7", "c","u", "]", la caja resultante se asemeja a la correspondiente *upper case letter*. Un valor de "n" (n entre comillas) suprime la caja.
- cex: Corresponde a un valor numérico para especificar la cantidad por la cual el texto y los símbolos gráficos deberán ser escalados en relación al valor establecido por defecto.
- cex.axis: Para especificar la magnificación que debe ser usada para la anotación sobre los ejes (los valores en los ticks) en relación al valor actual.
- cex.lab: Para especificar la magnificación a ser usada en las etiquetas de los ejes x e y en relación al valor actual.
- cex.main: Para especificar la magnifiación a ser usada en los títulos principales de los gráficos en relación al valor actual.
- cex.sub: Para especificar la magnificación a ser usada en los subtítulos en relación al valor actual.

- cin: R.O.; Tamaño de caracter "(ancho,alto)" en pulgadas.

- col: Para especificación del color de los símbolos plotting a ser usados. A specification for the default plotting color. Los colores pueden ser especificados de las siguientes formas: con una cadena de caracteres dando el nombre del color (en inglés) entre comillas, por ejemplo "blue", Una lista de los posibles colores puede obtenerse con la función "colors". Otra manera de especificar colores es en términos de componentes RGB con una cadena de la forma "#RRGGBB" en donde cada uno de los pares RR, GG, BB consiste de dos dígitos hexadecimales dando un valor en el rango de 00 a FF. Los colores también pueden especificarse dando un índice de una pequeña tabla de colores lo cual es compatible con S. También las funciones "rgb", "hsv", "gray" y "rainbow" proporcionan formas adicionales para generar colores.

- col.axis: Para especificar a ser usado para la anotación de los ejes (los valores en los ticks)

- col.lab: Para especificar el color a ser usado en las etiquetas de los ejes x e y.

- col.main: El color a ser usado en los títulos principales de los gráficos.

- col.sub: El color a se usado en los subtítulos.

- cra: R.O.; Tamaño de caracter "(ancho,alto)" en "rasters" (pixels).

- crt: Un valor numérico para especificar en grados como deben ser rotados los caracteres individuales.

  Es imprudente esperar que valores diferentes a los múltiplos de 90 funcionen

- csi: R.O.; Para especificar la altura en pulgadas de los caracteres.

- cxy: R.O.; Tamaño "(ancho,alto)" de caracter por defecto en *user coordinate units*. **par("cxy")** es **par("cin")/par("pin")** escalado a las coordenadas de usuario. c(strwidth(ch),strheight(ch)) para una cadena ch es usualmente mucho más precisa.

- din: R.O.; Dimensiones del dispositivo en pulgadas.

- err: Para el grado de reporte de error deseado; sin embargo no está implementado; R no avisa cuando puntos por fuera de la región de gráfico no son graficados.
- fg: El color a ser usado para el primer plano de los gráficos. Es el color que por defecto es usado en objetos como los ejes y las cajas alrededor de los gráficos.
- fig: Un vector numérico de la forma c(x1, x2, y1, y2) el cual da las coordenadas (NDC) de la región de la figura en la región de presentación del dispositivo.
- fin: Un vector numérico de la forma c(x, y) el cual da el tamaño de la región de la figura en pulgadas.
- font: Un entero que especifica la fuente a usar para el texto, de modo que:
  - 1 corresponde a texto plano,
  - 2 corresponde a texto en negrilla,
  - 3 corresponde a texto en itálica, y
  - 4 corresponde a texto itálica - negrilla
- font.axis: Para especificar la fuente a ser usada en la anotación de los ejes
- font.lab: Para especificar la fuente a ser usada en las etiquetas de los ejes x e y.
- font.main: Para especificar la fuente a ser usada en los títulos principales de los gráficos.
- font.sub: Para especificar la fuente a ser usada en los subtítulos.
- gamma: Para la corrección gamma, ver hsv(.., gamma) más adelante.
- lab: Un vector numérico de la forma c(x, y, len), el cual modifica la forma en que los ejes son anotados. Los valores de x e y dan el número aproximado de *tickmarks* en los respectivos ejes y *len* especifica el tamaño de etiqueta. Por defecto corresponde a c(5, 5, 7). Sin embargo, actualmente *len* no está implementado.

- las: Un número en $\{0, 1, 2, 3\}$, con el cual se especifica el estilo de las etiquetas de las anotaciones en los ejes:
  - 0: Siempre paralelo al eje (valor por defecto),
  - 1: siempre horizontal,
  - 2: siempre perpendicular al eje,
  - 3: siempre vertical.

  Notar que la rotación de cadenas de caracteres o de caracteres individuales (via par(srt = ..)) no afecta las etiquetas de ejes.

- lty: Para especificar el tipo de linea. Los tipos pueden especificarse con un número entero (0=blank (linea invisible), 1=solid, 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) o mediante uno de las siguientes cadenas de caracteres (entre comillas): "blank", "solid", "dashed", "dotted", "dotdash", "longdash", o "twodash".
  También es posible especificar los tipos usando una cadena de hasta 8 caracteres de c(0:9,"A":"F"), dando la longitud de los segmentos de linea en posiciones consecutivas en la cadena, los cuales son alternativamente dibujados y ?. Por ejemplo, la cadena "33" especifica 3 unidades "on" seguidas de 3 unidades "off" seguidas por 1 "on" y finalmente 3 "off". Las unidades son proporcionales a "lwd", y con "lwd=1" corresponden a pixels o puntos.

- lwd: Un número positivo para determinar el ancho de linea. Por defecto es 1.

- mai: Un vector numérico de la forma c(abajo, izquierda, arriba, derecha) que da los tamaños de las márgenes especificados en pulgadas.

- mar: Un vector numérico de la forma c(abajo, izquierda, arriba, derecha) el cual da las lineas de las márgenes a ser especificadas sobre los cuatro lados del gráfico. Por defecto es $c(5, 4, 4, 2) + 0.1$.

- mex: Es un factor de expansión de tamaño de caracter el cual es usado para describir las coordenadas en las márgenes de los gráficos.

- mfcol, mfrow: Un vector de la forma A vector c(nr, nc). Los gráficos subsecuentes serán dibujados en un arreglo nr -por- nc en el dispositivo

por columnas (mfcol, con el cual los gráficos sucesivos se van acomodando por columnas de izquierda a derecha y de arriba hacia abajo), o por filas (mfrow, con el cual los gráficos sucesivos se van acomodando por fila, de izquierda a derecha y de arriba hacia abajo), respectivamente. Alternativas a este argumento son layout(...) y split.screen(...).

- mfg: Un vector numérico de la forma c(i, j) donde i y j indican cual figura en un arreglo de figuras va a ser dibujada a continuación o está siendo dibujada. El arreglo debe haber sido previamente ajustado por par(mfcol=c(..)) o par(mfrow=c(..)).

- mgp: La linea de margen en unidades "mex" para el título de ejes, las etiquetas de ejes y lineas de ejes. Por defecto es c(3, 1, 0).

- mkh: Para especificar la altura en pulgadas de los símbolos a ser dibujados cuando el valor de "pch" es un entero. Actualmente éste es completamente ignorado.

- new: Argumento lógico, por defecto es igual a FALSE. Si se ajusta a TRUE, el siguiente comando de graficación de alto nivel no limpiará el marco o ventana antes de dibujar, es decir superpone el nuevo gráfico al inmediatamente anterior.

- oma: Un vector de la forma c(abajo, izquierda, arriba, derecha) que da el tamaño de las márgenes exteriores en lineas de texto.

- omd: Un vector de la forma c(x1, x2, y1, y2) para especificar la región de margen externa en NDC (= normalized device coordinates), es decir, como fracción (en $[0, 1]$) de la región de dispositivo.

- omi: Un vector de la forma c(abajo, izquierda, arriba, derecha) para dar el tamaño de las márgenes exteriores en pulgadas.

- pch: Puede ser un entero para especifica un símbolo o un sólo caracter a ser usado por defecto en la graficación de puntos.

- pin: Un vector de la forma c(x,y) para especificar el ancho y alto del gráfico actual en pulgadas.

- plt: Un vector de la forma c(x1, x2, y1, y2) que da las coordenadas de la región de gráfico como fraccide la actual región de figura.

- ps: Un número entero con el cual se determina el tamaño de los puntos en texto y símbolos.
- pty: Un caracter entre comillas para especificar el tipo de región gráfica a ser usada:
  - "s" genera una región de graficación cuadrada,
  - "m" genera una región de graficación máxima.
- srt: Rotación en grados de una cadena de caracteres.
- tck: Para especificar la longitud de los *tick marks* como una fracción del valor más pequeño entre el ancho y alto de l región de graficación. Si tck=1, una rejilla es dibujada. El ajuste por defecto es usar tcl=-0.5.
- tcl: La longitud de los *tick marks* como una fracción de la altura de un linea de texto. Por defecto es -0.5.
- tmag: Un número para especificar el ensanchamiento del texto del título principal en relación al otro texto de anotación del gráfico.
- type: Se especifica como un caracter entre comillas para determinar el tipo de gráfico. Por defecto es "p". Ver plot.default(type=...).
- usr: Un vector de la forma c(x1, x2, y1, y2) para especificar los extremos de las coordenadas del usuario de la región de graficación. Cuando se usa un escala logarítmica, por ejemplo par("xlog"), en este caso entonces los límites-x serán

```
10 ^ par("usr")[1:2]
```

- xaxp: Un vector de la forma c(x1, x2, n) que da las coordenadas de los *tick-marks* extremos y el número de intervalos entre *tick-marks* para el eje x.
- xaxs: El estilo de cálculo de intervalo de eje para el eje x. Sus valores posibles son (entre comillas):
  - "r", regular, que extiende el rango de datos por 4% y entonces halla un eje con etiquetas regulares (igualmente espaciadas) (*pretty labels*) que ajuste dentro del rango.

  - "i", interno, sólo halla un eje con etiquetas regulares que ajuste dentro del rango original de los datos.
  - "s", standard, halla un eje con etiquetas regulares dentro del cual ajuste el rango de los datos originales.
  - "e", extendido, similar al estilo "s", pero además se asegura que haya espacio para los símbolos de graficación dentro de la caja limitante.
  - "d", directo, especifica que el eje actual sea usado en los gráficos subsecuentes.

  De los anteriores estilos sólo "r" y "i" están actualmente implementados.

- xaxt: Un caracter que permite especificar el tipo de eje para el eje x. "n" hace que el eje sea ajustado pero no graficado. El valor estándar es "s": Para compatibilidad con S los valores "l" y "e" son aceptados pero son equivalentes a "s".

- xlog: R.O.; Valor lógico. Si es TRUE, una escala logarítmica será usada en el eje x, pero en una función como plot es necesario el siguiente argumento: plot(...,log="x",...). Para un nuevo dispositivo (cuando se cierra la ventana gráfica y luego se genera un nuevo gráfico se tiene un nuevo dispositivo), por defecto es FALSE, es decir, se ajusta la escala lineal.

- xpd: Puede ser un valor lógico o NA. Si es FALSE, todo *plotting* es sujeto a la región de graficación, si es TRUE todo *plotting* es sujeto a la región de figura y si es NA, todo *plotting* es sujeto a la región de dispositivo.

- yaxp: Un vector de la forma c(y1, y2, n) que da las coordenadas de los *tick-marks* extremos y el número de intervalos entre *tick-marks* para el eje y.

- yaxs: El estilo de cálculo de intervalo de eje para el eje y. ver xaxs arriba.

- yaxt: Un caracter que permite especificar el tipo de eje para el eje y. "n" hace que el eje sea ajustado pero no graficado.

- ylog: R.O.; un valor lógico, Si es TRUE, una escala logarítmica será usada en el eje y, pero en una función como plot es necesario el siguiente argumento: plot(...,log="y",...). Para un nuevo dispositivo (cuando se cierra la ventana gráfica y luego se genera un nuevo gráfico se tiene un nuevo dispositivo), por defecto es FALSE, es decir, se ajusta la escala lineal.

Notas:

1. Los parámetros son consultados dando uno o más vectores de caracteres para par.

2. par() (sin argumentos) o par(no.readonly=TRUE) se usa para obtener todos los parámetros gráficos. Sus nombres son tomados de la variable '.pars'.

3. '.Pars.readonly' contiene los nombres de los argumentos de par que son "readonly".

4. Argumentos R.O. := Read-only arguments: estos sólo pueden usarse en *queries*, es decir, no ajustan nada.

5. Todos los anteriores argumentos, excepto los R.O. y los siguientes argumentos "low-level", pueden ser ajustados también en funciones de graficación "high-level" y "mid-level", tales como plot, points, lines,axis, title, text, mtext:

    - ask
    - fig, fin
    - mai, mar, mex
    - mfrow, mfcol, mfg
    - new
    - oma, omd, omi
    - pin, plt, ps, pty
    - usr
    - xlog, ylog

6. Cuando los parámetros son ajustados, sus valores anteriores son restituídos en una lista *named* invisible. Tal lista puede ser pasada como un argumento a par para restaurar los valores de los parámetros. Usar par(no.readonly = TRUE) para que la lista completa de parámetros pueda ser restaurada.

7. Cuando sólo un parámetro es consultado, el valor es una cadena caracter. Cuando dos o más parámetros son consultados, el resultado es una lista de cadenas de caracter, con los nombres de lista dando los parámetros.

   Notar la inconsistencia: Ajustar un parámetro retorna una lista, pero consultar (*querying*) un parámetro retorna un vector.

8. Es difícil predecir el efecto de restaurar todos los parámetros gráficos (ajustables) si el dispositivo ha sido redimensionado. Varios de ellos intentan ajustar lo mismo de diferentes formas, y ganarán los últimos en el alfabeto. En particular, los ajustes de mai, mar, pin, plt y pty interactúan, como hacen los ajustes de márgenes exteriores, el diseño de figura y el tamaño de la región de figura.

7.2 Especificación de color (función `hsv`)

hsv es una función que crea un vector de colores especificando matiz, saturación y valor:

```
hsv(h=1, s=1, v=1, gamma=1)
```

Argumentos:

- h,s,v: Vectores numéricos de valores en el rango $[0, 1]$ para que "hue", "saturation" y "value" sean combinados para formar un vector de colores correspondiente a los valores dados en espacio HSV. Los valores en argumentos más cortos son reciclados. Los valores retornados por hsv pueden usarse con una especificación 'col= ' en funciones gráficas o en par.

- gamma: Una "corrección gamma".

Ejemplos:

```
hsv(.5,.5,.5)
[1] "#408080" #este es el valor que devuelve hsv(.5,.5,.5)

plot(rnorm(10),col=hsv(.5,.5,.5),pch=19)
plot(rnorm(10),col="#408080",pch=19)

#ver efecto gamma (intente lo siguiente):
n <- 20

y <- -sin(3*pi*((1:n)-1/2)/n)

op <- par(mfrow=c(3,2),mar=rep(1.5,4))

for(gamma in c(.4, .6, .8, 1, 1.2, 1.5))

plot(y, axes = FALSE, frame.plot = TRUE,
xlab = "", ylab = "", pch = 21, cex = 30, bg =
rainbow(n, start=.85, end=.1, gamma = gamma),
main = paste("tonos rojos;
gamma=",format(gamma)))

par(op)
```

7.3 Función `curve`

Esta función dibuja la curva correspondiente a una función o expresión dada en x sobre el intervalo $[from, to]$:

```
curve(expr, from, to, n = 101, add = FALSE, type = "l",
ylab = NULL, log = NULL, xlim =
NULL, ...)

plot.function(fn, from = 0, to = 1, xlim, ...)
```

Los argumentos son:

- expr: Es una expresión escrita como una función de x, o alternativamente una función la cual será graficada.
- fn: Una función R numérica vectorizante.
- from,to: El rango sobre el cual la función será graficada.
- n: Un entero que representa el número de valores x en los cuales se evaluará la función.
- add: Argumento lógico. Si es TRUE adiciona la curva a un gráfico ya existente.
- xlim: Argumento numérico de longitud 2; si se especifica, sirve como defecto para 'c(from, to)'
- ...: Parámetros gráficos también pueden ser especificados como argumentos.
- 'plot.function' Pasa todos estos a 'curve'.

Detalles: 'expr' es evaluado en 'n' puntos igualmente espaciados sobre el rango '$[from, to]$'. Los puntos determinados de esta forma son unidos por lineas rectas. 'fn(x)' o 'expr' (como función de x) debe retornar un arreglo numérico de la misma longitud de x.

Si 'add=TRUE', 'c(from,to)' toma por defecto 'xlim' el cual toma por defecto los límites x del gráfico actual. Además 'log' es tomado del gráfico actual cuando 'add=TRUE'.

Esta función puede dar malos resultados para graficar curvas que no son suaves.

Ejemplo 1:

```
op <- par(mfrow=c(2,2))

curve(x^3-3*x, -2, 2,ylab=expression(f(x)),
main=expression(paste(x^3-3*x,sep="     ",'y',sep=" ",x^2-2)),
cex.main=0.9) curve(x^2-2,
add = TRUE, lty=2,cex.main=0.9)

plot(cos, xlim = c(-pi,3*pi), n =
1001,main=expression(cos(x)),ylab=expression(f(x)),
cex.main=0.9)

fx<- function(x) sin(cos(x)*exp(-x/2))

curve(fx, -8, 7,
n=2001,main=expression(sin(cos(x)*exp(-x/2))),
ylab=expression(f(x)),cex.main=0.9)

curve(fx, -8,
-5,main=expression(sin(cos(x)*exp(-x/2))),
ylab=expression(f(x)),cex.main=0.9)

par(oma=c(1,1,1,1),font=2)

mtext(outer=T,"Gráficos de funciones con 'curve' ",side=3)

par(op)
```

Ejemplo 2: Uso de escalas logarítmicas:

```
op <- par(mfrow=c(2,2))

for(i in c("","x","y","xy"))

curve(log(1+x), 1,100, log=i,
sub=paste("log= '",i,"'",sep=""))
```

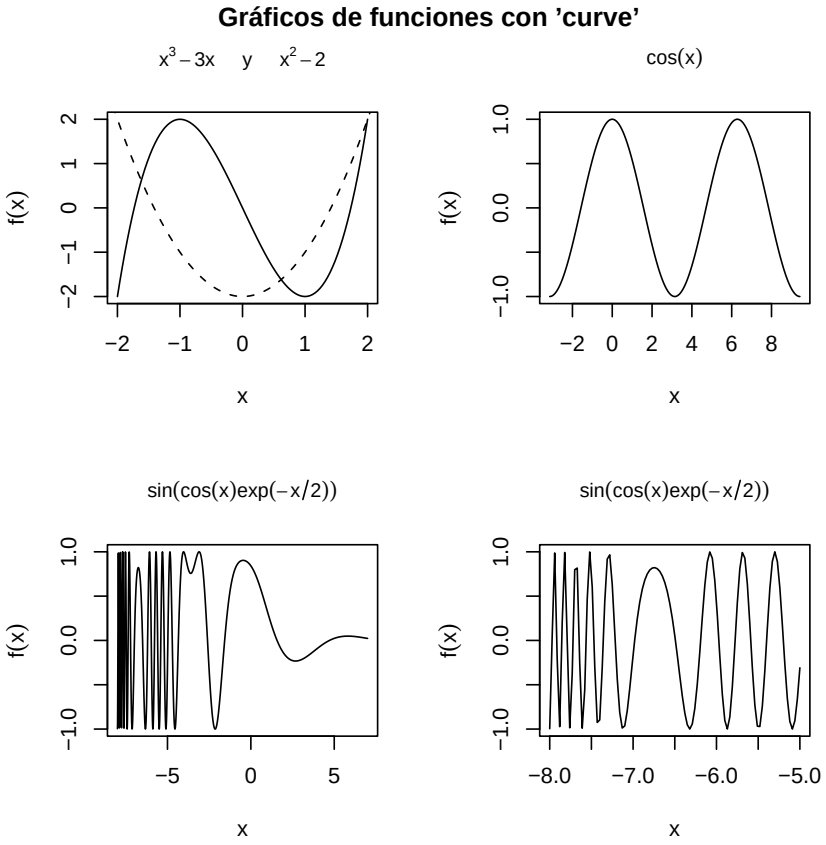
Gráficos de funciones con 'curve'
x^3 − 3x y x^2 − 2
cos(x)
sin(cos(x)exp(−x/2))
sin(cos(x)exp(−x/2))
f(x)
x

Figura 7.1:

```
par(oma=c(1,1,1,1),font=2)

mtext(outer=T,"Gráficos de funciones con 'curve'
y efecto de 'log' ",side=3)

par(op)
```

7.4 Función `points`

Esta función permite agregar puntos a un gráfico previo, luego sólo funciona si se ha llamado algún gráfico antes de invocar esta función. Dibuja la secuencia de puntos (usando el tipo de *plotting character* especificado con el argumento 'pch=') de acuerdo a las coordenadas especificadas:

```
points(x, ...) points.default(x, y=NULL, type="p",
pch=1, col="black", bg=NA, cex=1, ...)
```

Argumentos:

- x, y:Vector de coordenadas de los puntos a graficar.
- type: Caracter que identifica el tipo de gráfico, que corresponde a los mismos disponibles para la función 'plot(...)'
- pch: símbolo a usar para representar a los puntos. Puede especificarse como un caracter entre comillas o un código entero correspondiente a un conjunto de símbolos gráficos. El conjunto total de estos últimos están disponibles de 0 a 25, de los cuales los correspondientes los enteros 21 a 25 pueden ser coloreados y rellenado con colores diferentes:
  - 'pch=19': círculo sólido
  - 'pch=20': Bullet (círculo más pequeño)
  - 'pch=21': círculo
  - 'pch=22': Cuadrado

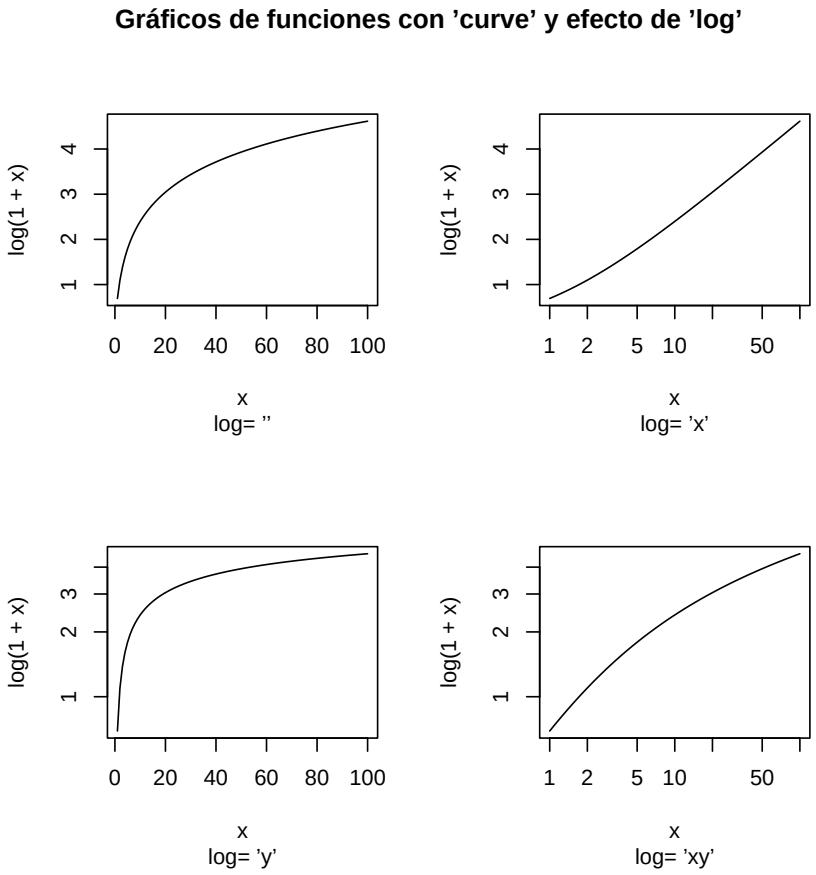

Figura 7.2: *Los efectos gráficos del uso de una escala logarítmica en uno o en ambos ejes puede apreciarse en estos gráficos. En el gráfico de la esquina izquierda arriba, no hay uso de la escala logarítmica; en el siguiente a la derecha, los valores en x han sido transformados a una escala logarítmica es decir, cada unidad en dicho eje corresponde al log del valor señalado; en el gráfico de la esquina inferior izquierda se ha hecho uso de una escala logarítmica en el eje y de modo que las unidades en éste corresponden al log del valor señalado. Finalmente, en el gráfico de la esquina inferior derecha se ha hecho uso de una escala logarítmica en ambos ejes.*

  - 'pch=23': Diamante
  - 'pch=24': Triángulo punta hacia arriba
  - 'pch=25': Triángulo punta hacia abajo

- col: Código o nombre (entre comillas) del color. Ver 'colors', 'palette'.
- bg: Color de relleno para símbolos de graficación abiertos.
- cex: Factor de expansión de caracter (define tamaño del símbolo).
- ...: Parámetros gráficos adicionales (ver 'plot.xy' y 'par') pueden ser dados como argumentos.

Detalles: Las coordenadas pueden ser proporcionadas en una estructura de graficación (lista con componentes x e y), en una matriz de dos columnas, en una serie de tiempo. Ver 'xy.coords' (función para extracción de estructuras de graficación).

Ejemplo 1:

```
# Ajustar un sistema de coordenadas:
plot(-4:4, -4:4, type = "n",xlab="x",ylab="y",
main="Uso de la función 'points' ")

#graficar los puntos deseados:
points(rnorm(200), rnorm(200), pch=23,bg = "gray")
points(rnorm(100)/2, rnorm(100)/2, pch=22, cex = 1.5)
```

Ejemplo 2: Presentación de todos lo símbolos gráfico y otros adicionales.

```
Pex <- 3
ipch <- 1:(np <- 25+11)
k <- floor(sqrt(np))
dd <- c(-1,1)/2
rx <- dd + range(ix <- (ipch-1) %/% k)
ry <- dd + range(iy <- 3 + (k-1)-(ipch-1) %% k)
```

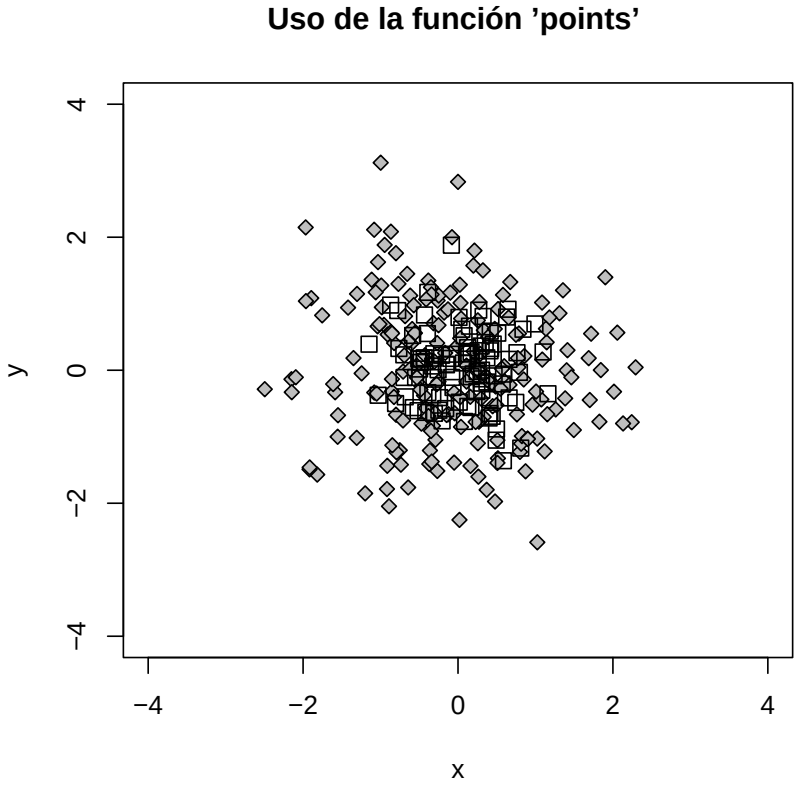

Figura 7.3: *El uso de diferentes símbolos para representación de puntos puede resultar útil para identificar ciertos agrupamientos o estructuras en los datos.*

```
pch <- as.list(ipch)
pch[25+ 1:11] <- as.list(c("*",".", "o","O",
"0","+","-",":","|","%","#"))

plot(rx, ry, type="n", axes = FALSE, xlab = "", ylab = "",
main = paste("Símbolos de
graficación:\npoints (..  pch = *, cex =", Pex,")"))

box()

abline(v = ix, h = iy, col = "black", lty = "dotted")

for(i in 1:np) {
pc <- pch[[i]]

#los símbolos aparecer\'an con
#perfil negro y/o interior azul,

#en los casos que sea factible:

points(ix[i], iy[i], pch = pc, col = "black", bg = "blue",
cex = Pex) text(ix[i] - .4,
iy[i], pc, col = "brown", cex = 1.5) }
```

7.5 Función `lines`

Esta función permite graficar segmentos de lineas rectas en un gráfico previo:

```
lines(x, ...) lines.default(x, y=NULL, type="l",
col=par("col"), lty=par("lty"), ...)
```

Argumentos:

- x, y: Vectores de coordenadas de los puntos a unir.

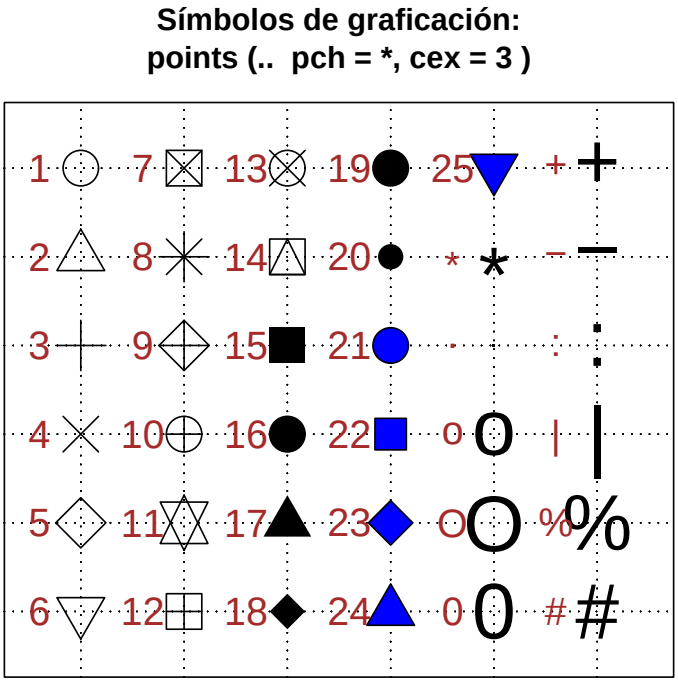

Figura 7.4: *Los números del 1 al 25 al lado derecho corresponden al código numérico con el cual se obtiene el correspondiente símbolo. Los restantes símbolos se obtienen especificando pch igual al correspondiente caracter entre comillas.*

- type: Caracter que indica el tipo de gráfico; pueden especificarse los tipos plausibles en el argumento 'type' de la función 'plot(...)'.
- col: Color a usar.
- lty: Tipo de linea a usar.
- ...: Pueden especificarse otros parámetros gráficos (ver 'par'), tales como 'lwd'.

Detalles: Las coordenadas pueden suministrarse a 'lines' en una estructura de graficación (una lista con componentes x e y), etc. Ver 'xy.coords'.

Las coordenadas pueden contener valores 'NA'. Cuando esto sucede el punto correspondiente es omitido del gráfico, y ninguna linea es dibujada hasta o desde tal punto. Por tanto, pueden usarse valores faltantes para obtener discontinuidades en las lineas.

Ejemplo 1:

```
y<-rnorm(12)
x<-1:12
plot(x,y,xaxt="n",cex.axis=0.8,pch=23,bg="gray",
col="black",cex=1.1,main="Uso de 'lines'
para dibujar una serie",cex.main=0.9)
axis(1,at=1:12,lab=month.abb,las=2,cex.axis=0.8)
lines(x,y,lwd=1.5)
```

Ejemplo 2:

```
x<-rnorm(50)
y<-2*x+rnorm(50)
library(modreg)
plot(x,y,main="Ejemplo de la función
'lines'\ndibujando una curva suavizada")
lines(loess.smooth(x,y))
```

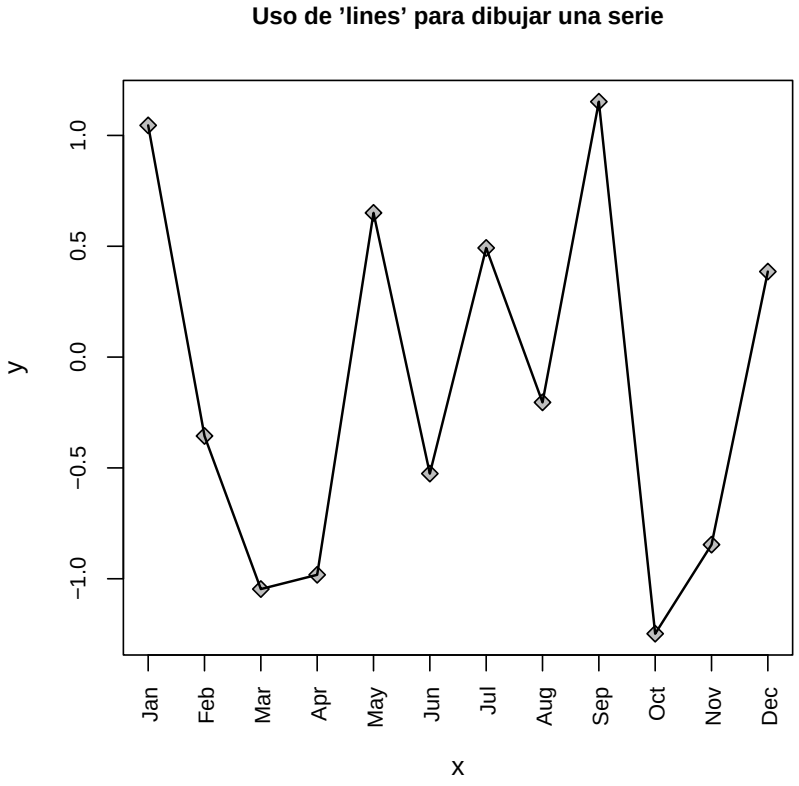

Figura 7.5: *'lines' permite agregar líneas a un gráfico, lo cual es útil pues a pesar que la función plot posee la opción 'type="l"', sólo con 'lines' es posible trazar segmentos de recta particulares. En este ejemplo vemos una aplicación.*

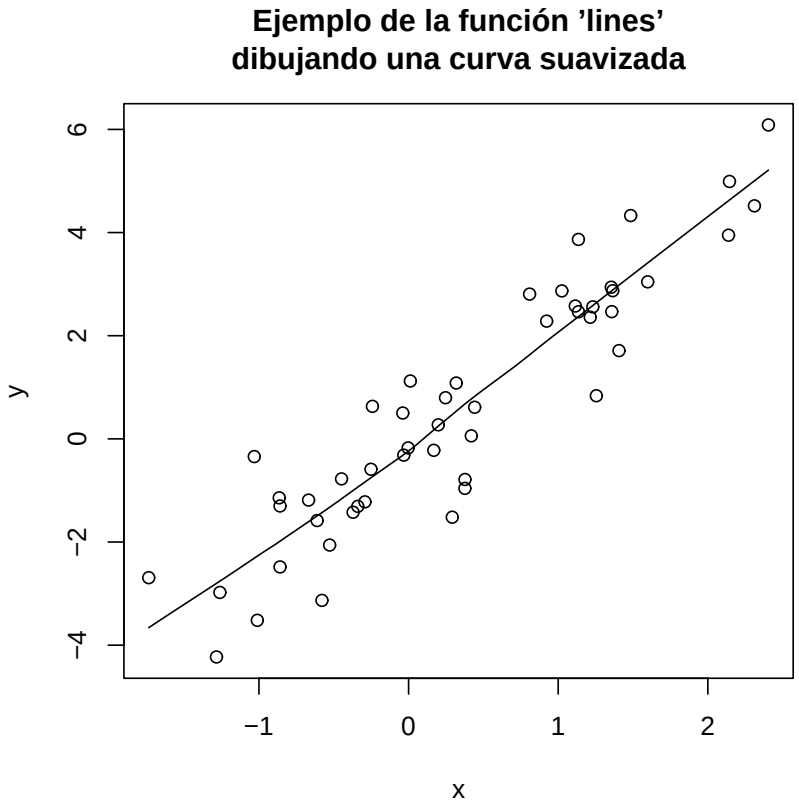

Figura 7.6: *Además de permitir la adición de segmentos de recta a un gráfico, 'lines' prueba su utilidad en la graficación de curvas suavizadas provenientes de ajustes no paramétricos, en este caso un suavizamiento loess.*

7.6 Función `abline`

Al igual que 'lines' esta función permite una o más lineas rectas a un gráfico:

```
abline(a, b, untf = FALSE,\ dots)

abline(h=, untf = FALSE, ...)

abline(v=, untf = FALSE,...)

abline(coef=, untf = FALSE, ...)

abline(reg=, untf = FALSE, ...)
```

Argumentos:

- a,b: El intercepto y la pendiente de la recta.
- untf: Argumento lógico. Ver detalles.
- h: El valor y para un recta horizontal $y = f(x) = c$. En este caso, a y b no se especifican
- v: El valor x para una recta vertical $x = f(y) = k$.
- coef: Un vector de longitud 2 que da el intercepto y la pendiente.
- reg: un objeto con un componente 'coef'. Ver detalles.
- ...: graphical parameters.

Detalles:

- La recta puede especificarse de dos formas. La primera especifica la linea en forma de intercepto/pendiente. La otra forma es especificar a 'a' como un vector, que contiene el intercepto y la pendiente.
- Los argumentos 'h=' y 'v=' dibujan lineas verticales y horizontales respectivamente en las coordenadas especificadas.

- El argumento 'coef' especifica la linea por un vector que contiene la pendiente y el intercepto.
- 'reg' es un objeto de regresión que contiene $'reg\$coef'$. Si es de longitud 1 entonces el valor disponible se toma como la pendiente de la recta a través del origen, de lo contrario, los primeros dos valores son tomados como el intercepto y la pendiente.
- If 'untf' es TRUE, y uno o ambos ejes están en escala con transformación log, entonces una curva es dibujada correspondiendo a una linea en las coordenadas originales, de lo contrario una linea es dibujada en el sistema de coordenadas transformado. los parámetros 'h' y 'v' siempre se refieren a las coordenadas originales.

ver también 'lines' y 'segments'

Ejemplo 1: Trazado de una linea de regresión

```
bivariada<-function(n,mu.x,mu.y,sigma.x,sigma.y,ro){
x<-c(rnorm(n,mu.x,sigma.x))
mu.y.x<-mu.y+ro*sigma.y*(x-mu.x)/sigma.x
sigma.y.x<-sigma.y*sqrt(1-ro^2)
y<-c(rnorm(n,mu.y.x,sigma.y.x))
datos<-cbind(x,y) datos
}

datos<-bivariada(50,2,3,2,3,0.9)

x<-datos[,1]
y<-datos[,2]

#Regresion lineal simple de y en x:
regres<-lm(y~x)
regres
Call: lm(formula = y ~ x)

Coefficients:
(Intercept)           x
```

```
     0.3652       1.3773

#Grafica de dispersion con
#linea de regresion:

plot(x,y,pch=19,main="Uso de 'abline'
para trazar linea de regresión\nsobre gráfico de
dispersión",cex.main=0.95)

abline(regres,lwd=1.5)

#O bien:

plot(x,y,pch=19,main="Uso de 'abline' para trazar
linea de regresión\nsobre gráfico de
dispersión",cex.main=0.95)

abline(reg=regres,lwd=1.5)

#O bien:

plot(x,y,pch=19,main="Uso de 'abline' para trazar
linea de regresión\nsobre gráfico de
dispersión",cex.main=0.95)

abline(a=regres,lwd=1.5)
```

Ejemplo 2: Trazado de lineas horizontales y verticales

```
y<-rnorm(50)
x<-rnorm(50)

plot(x,y,pch=19,xlim=c(-4,4),ylim=c(-4,4),
main="Uso de 'abline' para trazar linea
horizontal y vertical en un gráfico",cex.main=0.95)
abline(h=0,lty=2,lwd=1.5)
```

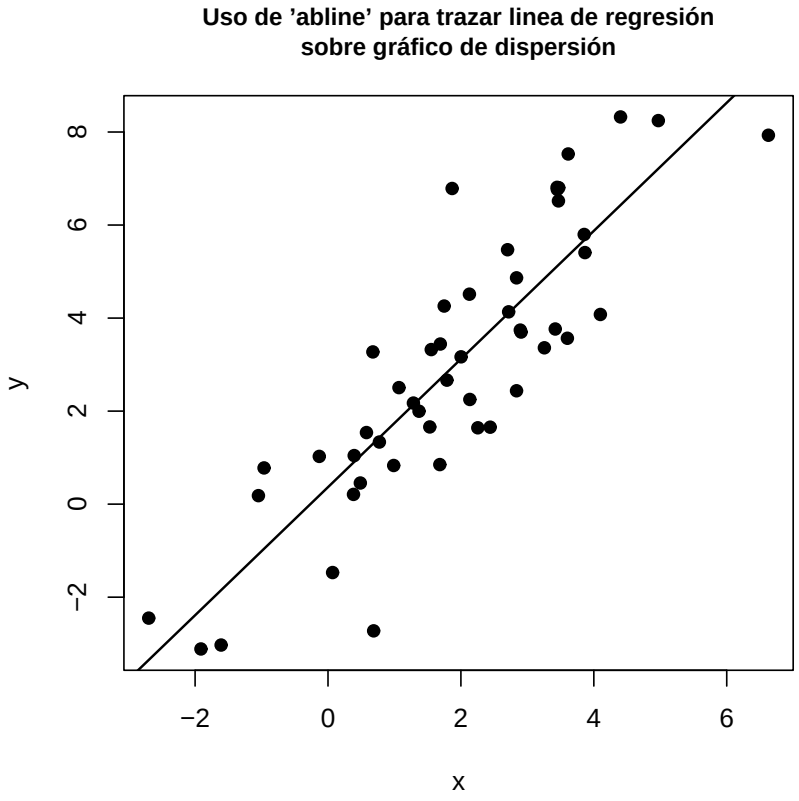

Figura 7.7: *A diferencia de 'lines', 'abline' sólo grafica líneas rectas con diferente pendiente e intercepto. En la figura, la recta ajustada en una regresión simple ha sido obtenida aplicando 'abline' sobre un objeto R de clase 'lm'*

```
abline(v=0,lty=2,lwd=1.5)
```

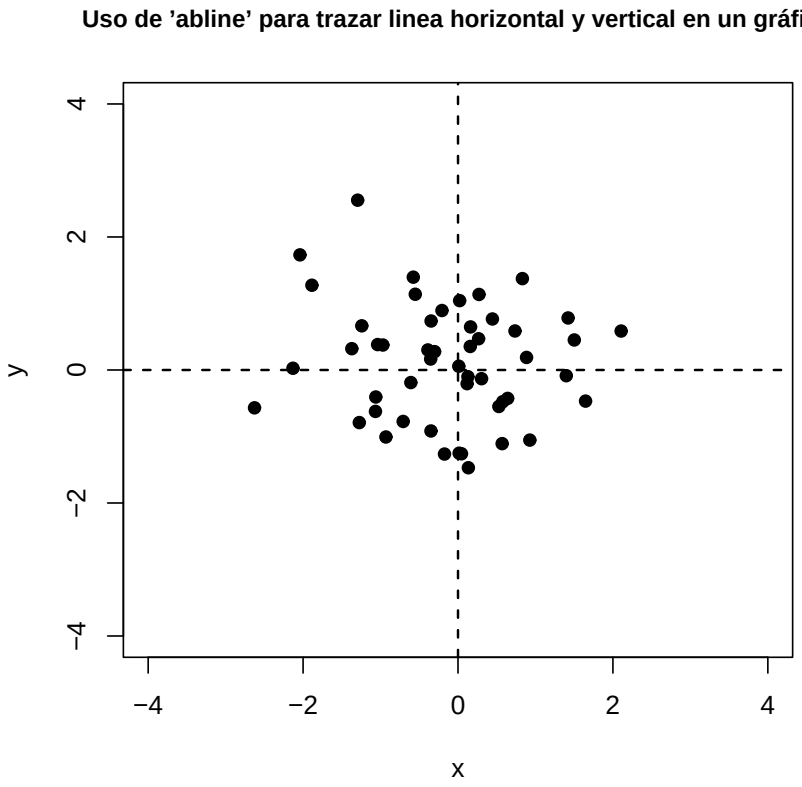

Figura 7.8: *Líneas de referencia en un gráfico pueden resultar útiles para identificar ciertas propiedades o puntos que cumplan ciertas condiciones. Para el caso las líneas punteadas facilitan identificar el centro de los puntos de datos, la dispersión alrededor de éste y los posibles puntos extremos.*

7.7 Función `contour`

Esta funcón permite realizar gráficos de contornos, tales como los contornos de verosimilitud en dos parámetros o también en la graficación de regiones de

confianza, mapas topográficos, etc. Dichos contornos pueden ser agregados a un gráfico previo, por ejemplo a un gráfico de dispersión:

```
contour(x = seq(0, 1, len = nrow(z)),
y = seq(0, 1, len = ncol(z)), z,nlevels = 10,
levels = pretty(zlim, nlevels), labels = NULL,
xlim = range(x, finite = TRUE),ylim =
range(y, finite = TRUE), zlim = range(z, finite = TRUE),
labcex = 0.6, drawlabels = TRUE,
method = "flattest",vfont = c("sans serif", "plain"),
axes = TRUE, frame.plot = axes, col
= par("fg"), lty = par("lty"), lwd = par("lwd"),
add = FALSE, ...)
```

Argumentos:

- x,y: Ubicación de las lineas de la rejilla sobre la cual los valores en 'z' son medidos. éstas deben estar en orden ascendente. Por defecto, se usan valores igualmente espaciados en el intervalo de 0 a 1. Si x es una lista, sus componentes $x\$x$ y $x\$y$ son usados para x e y respectivamente. Si la lista tiene componente 'z' ésta es usada para 'z'.

- z: Una matriz que contiene los valores a ser graficados (son permitidos valores 'NA'). Notar que 'x' puede ser usada en ves de 'z' por conveniencia.

- nlevels: El número de niveles de contorno deseados si 'levels' no es especificado.

- levels: Vector numérico que da los niveles en los cuales se desea dibujar las lineas de contorno.

- labels: Un vector que proporciona etiquetas para las lineas de contorno. Si es 'NULL' entonces los niveles son usados como etiquetas.

- labcex: Factor de expansión de caracter 'cex' para la etiquetación de los contornos.

- drawlabels: Argumento lógico. Si es 'TRUE' los contornos son etiquetados.

- method: cadena de caracteres que especifica dónde se ubicarán las etiquetas. Los valores posibles son (entre comillas) "simple", "edge" y "flattest" (por defecto). El primero dibuja la etiqueta en el borde del gráfico, sobreponiéndola a la linea de contorno; el segundo dibuja la etiqueta en el borde del gráfico , incrustado en la linea de contorno, sin superponerlas; y el tercero dibuja sobre la sección más plana del contorno, incrustando la etiqueta en la linea de contorno sin superposición. El segundo y tercer método pueden no dibujar un etiqueta en cada linea de contorno.

- vfont: Si se especifica un vector de caracteres de longitud 2, entonces *Hershey vector fonts* son usados para las etiquetas de los contornos. El primer elemento del vector selecciona un estilo o tipo de fuente y el segundo elemento selecciona un *fontindex*.

- xlim, ylim, zlim: Límites x, y y z para el gráfico.

- axes, frame.plot: Argumento lógico para indicar si ejes o una caja deberían ser dibujados.

- col: Color de las lineas.

- lty: Tipo de linea.

- lwd: Ancho de linea.

- add: Argumento lógico. Si es 'TRUE', adiciona los contornos a un gráfico.

- ...: Pueden suministrarse parámetros gráficos adicionales y argumentos para 'title'

Ver: 'filled.contour' para contornos con "color-filled", 'image' y 'demo(graphics).

Ejemplo 1: El log de verosimilitud de dos parámetros μ_1 y μ_2 es (ejemplo 10.2.1, Kalbfleisch. *Probability an statistical inference, vol.2, p.61*) es:

$$l(\mu_1, \mu_2) = -\frac{1}{2}(15.6 - \mu_1)^2 - \frac{1}{2}(29.3 - \mu_2)^2 - \frac{1}{2}(45.8 - \mu_1 - \mu_2)^2$$

Se desean graficar los contornos e intervalos de verosimilitud del 1, 10 y 50 %, sabiendo que:

$$l(\hat{\mu_1}, \hat{\mu_2}) = -0.135$$

$$r_{max}(\mu_1) = \frac{-3}{4}(\mu_1 - 15.9)^2 = -0.75\mu_1^2 + 23.85\mu_1 - 189.6075$$

$$r_{max}(\mu_2) = \frac{-3}{4}(\mu_2 - 29.6)^2 = -0.75\mu_2^2 + 44.4\mu_2 - 657.12$$

```
raiz.cuadratica<-function(a,b,c){
disc<-sqrt(b^2-4*a*c)
x1<-(-b-disc)/(2*a)
x2<-(-b+disc)/(2*a)
res<-c(x1,x2)
res
}
#Cálculo de intervalos de verosimilitud
#para mu1, del 1, 10 y 50%

a1<-0.75
b1<--23.85
aux1<-189.6075
c1.01<-aux1+log(0.01)
c1.10<-aux1+log(0.1)
c1.50<-aux1+log(0.5)
res1.01<-raiz.cuadratica(a1,b1,c1.01)
res1.01
[1] 13.42205 18.37795

res1.10<-raiz.cuadratica(a1,b1,c1.10)
```

```
res1.10
[1] 14.14783 17.65217

res1.50<-raiz.cuadratica(a1,b1,c1.50)
res1.50
[1] 14.93865 16.86135


a2<-0.75
b2<--44.4
aux2<-657.12
c2.01<-aux2+log(0.01)
c2.10<-aux2+log(0.1)
c2.50<-aux2+log(0.5)
res2.01<-raiz.cuadratica(a2,b2,c2.01)
res2.01
[1] 27.12205 32.07795


res2.10<-raiz.cuadratica(a2,b2,c2.10)
res2.10
[1] 27.84783 31.35217

res2.50<-raiz.cuadratica(a2,b2,c2.50)
res2.50
[1] 28.63865 30.56135

#Creacion de un grid en x,y
#y evaluacion de la funcion
#log de verosimilitud:

x<-seq(12,20,len=100)
y<-seq(26,33,len=100)

f<-outer(x,y,function(x,y)-0.5*(15.6-x)^2+
-0.5*(29.3-y)^2-0.5*(45.8-x-y)^2 +0.135)

#Graficacion de los contornos
```

```
#de verosimilitud del 1, 10 y 50%:

contour(x,y,f,levels=c(log(0.01),log(0.1),log(0.5)),
labels=c(0.01,0.1,0.5),lwd=1.5,xlab=expression(mu[1]),y
lab=expression(mu[2]),main="Contornos
de verosimilitud",cex.main=0.85)

#trazado de limites de intervalos de
#verosimilitud:

abline(v=res1.01,lty=2)
abline(v=res1.10,lty=2)
abline(v=res1.50,lty=2)
abline(h=res2.01,lty=2)
abline(h=res2.10,lty=2)
abline(h=res2.50,lty=2)

#etiquetado de lineas limites:

text(20,res2.01,labels=paste(round(res2.01,2)),pos=3,cex=0.6)
text(20,res2.10,labels=paste(round(res2.10,2)),pos=3,cex=0.6)
text(20,res2.50,labels=paste(round(res2.50,2)),pos=3,cex=0.6)
text(res1.01,26,labels=paste(round(res1.01,2)),pos=3,cex=0.6,
srt=90)
text(res1.10,26,labels=paste(round(res1.10,2)),pos=3,cex=0.6,
srt=90)
text(res1.50,26,labels=paste(round(res1.50,2)),pos=3,cex=0.6,
srt=90)

par(oma=c(1,1,3.5,1))
mtext(expression(l(mu[1],mu[2])==
-0.5*(15.6-mu[1])^2-0.5*(29.3-mu[2])^2-0.5*(45.8-mu[1]-mu[2])^2 ),
side=3,cex=0.85,outer=TRUE,font=2)
```

Ejemplo 2: Mapa topográfico (ejemplo disponible en el Help de R, paquete: base):

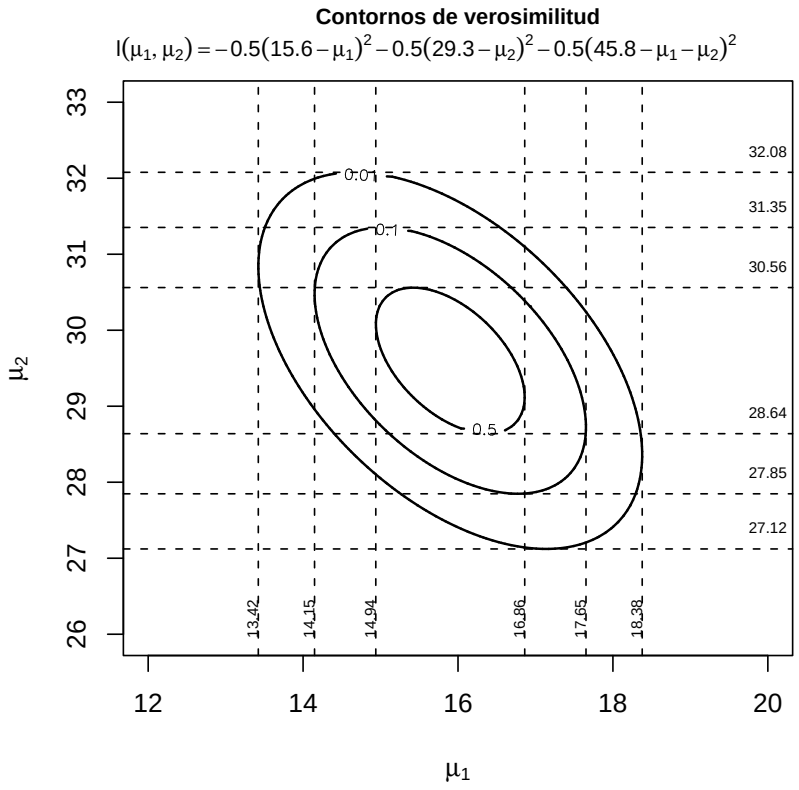

Figura 7.9: *'contour' permite graficar una función por niveles lo cual es útil en el trazo de contornos de verosimilitud, regiones de confianza, etc.*

```
data("volcano")
rx <- range(x <- 10*1:nrow(volcano))
ry <- range(y <- 10*1:ncol(volcano))
ry <- ry + c(-1,1) * (diff(rx) - diff(ry))/2

opar <- par(pty = "s")

plot(x = 0, y = 0,type = "n", xlim = rx, ylim = ry,
xlab = "", ylab = "")

u <- par("usr")

rect(u[1], u[3], u[2], u[4],  border = "black")

contour(x, y, volcano,lty = "solid", add = TRUE,
vfont = c("sans serif", "plain"))
title("Mapa topográfico de Maunga Whau", font = 4)

abline(h = 200*0:4, v = 200*0:4,lty = 2, lwd = 0.1)

par(opar)
```

7.8 Función `persp`

Esta función permite realizar gráficos tridimensionales de perspectiva de superficies sobre el plano $x - y$.

```
persp(x, ...)
persp.default(x = seq(0, 1, len = nrow(z)),
y = seq(0, 1, len = ncol(z)),
z, xlim = range(x), ylim = range(y),
zlim = range(z, na.rm = TRUE), xlab = NULL,ylab =
NULL, zlab = NULL, main = NULL, sub = NULL, theta = 0,
phi = 15, r = sqrt(3), d = 1,
scale = TRUE, expand = 1, col = NULL, border = NULL,
```

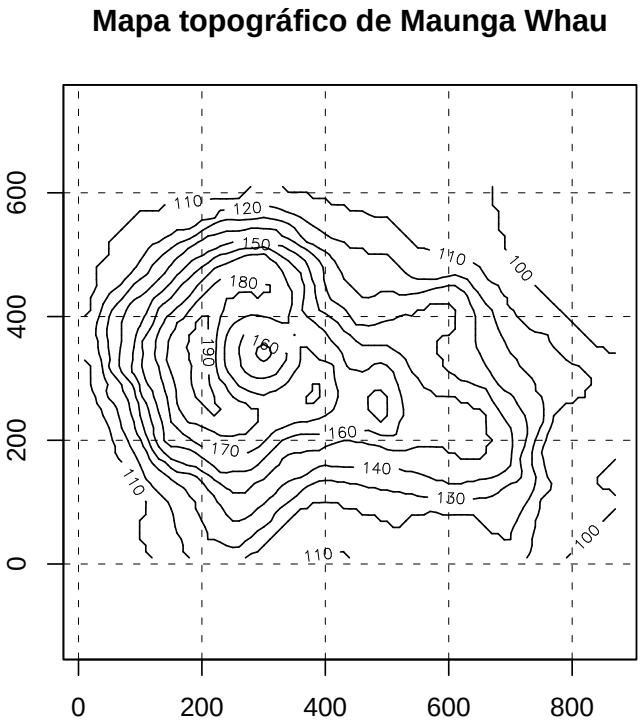

Figura 7.10: *Aquí observamos una aplicación no estadística pero bastante útil de la función 'contour' en el trazo de mapas topográficos.*

```
ltheta = -135, lphi = 0, shade = NA,
box = TRUE, axes = TRUE, nticks = 5, ticktype = "simple",...)
```

Argumentos:

- x, y: Ubicación de las lineas de la rejilla sobre la cual los valores en 'z' son medidos. éstas deben estar en orden ascendente. Por defecto, se usan valores igualmente espaciados en el intervalo de 0 a 1. Si x es una lista, sus componentes $x\$x$ y $x\$y$ son usados para x e y respectivamente.

- z: Una matriz que contiene los valores a ser graficados (valores 'NA' son permitidos). Notar que 'x' puede ser usada en ves de 'z' por conveniencia.

- xlim, ylim, zlim: Límites x, y y z para el gráfico. El gráfico es producido de manera que el volumen rectangular definido por estos límites sea visible.

- xlab, ylab, zlab: Títulos para los ejes. Estos deben ser cadenas de caracteres; no se permiten expresiones.

- main, sub: Título y subtítulo.

- theta, phi: Ángulos que definen la dirección de las vistas. 'theta' da la dirección azimutal y 'phi' la colatitud.

- r: La distancia del **eyepoint** desde el centro de la caja de graficación.

- d: Un valor que puede usarse para variar la longitud de la transformación de la perspectiva. Valores mayores que 1 reducirán el efecto perspectiva y valores menores o iguales a 1 lo exagerarán.

- scale: Antes de visualizar las coordenadas x, y y z de los puntos que definen la superficie éstas son transformadas al intervalo $[0, 1]$. Si 'scale' es TRUE las coordenadas son transformadas separadamente. Si es FALSE las coordenadas son escaladas de manera que el aspecto de las razones son mantenidas. Esto es útil para representar cosas como información DEM.

- expand: Un factor de expansión aplicado a las coordenadas z. A menudo es usado con 'expand' entre (0,1) para contraer el gráfico en la dirección z.

- col: El color de las facetas de la superficie.

- border: El color de la linea dibujada alrededor de las facetas de la superficie. Un valor de 'NA' inhabilitará el trazo de bordes. Esto resulta útil cuando la superficie es sombreada.

- ltheta, lphi: Si se especifican valores finitos para estos, la superficie es sombreada como si fuera iluminada desde la dirección especificada por el azimut 'ltheta' y l colatitud 'lphi'.

- shade: La sombra en un faceta de la superficie es calculada como $((1 + d)/2)^{shade}$, donde 'd' es el producto punto de un vector unitario normal a la faceta y un vector unitario en l dirección de un fuente de luz. Los valores de 'shade' cercanos a 1 producen sombreado similar a un modelo de fuente de luz puntual y valores cercanos a 0 no producen sombra. Valores en el rango de 0.5 a 0.75 proporcionan una aproximación a la iluminación diurna.

- box: Traza un caja para la superficie. Por defecto e 'TRUE'.

- axes: Agrega ticks y etiquetas a la caja. Por defecto es 'TRUE'. Si 'box' es 'FALSE' las etiquetas y los ticks no son dibujados.

- ticktype: Caracter: "simple" dibuja un flecha paralela al eje para indicar la dirección de aumento; "detailed" dibuja los ticks normales como en gráficos 2D

- nticks: El número aproximado de marcas ticks a dibujar en los ejes. No tiene efecto si 'ticktype' es "simple".

- ...: Parámetros gráficos adicionales.

Ejemplo 1: Graficación de normales bivariadas

```
x<-seq(-4,4,len=50)
y<-seq(-4,4,len=50)
```

```
normal.bivariada<-function(x,y,rho,mu1,sigma1,mu2,sigma2){
1/(2*pi*sigma1*sigma2*sqrt(1-rho^2))*exp(-1/(2*(1-rho^2))*
(((x-mu1)/sigma1)^2-2*rho*((x-mu1)/sigma1)*((y-mu2)/sigma2)+
((y-mu2)/sigma2)^2))
}

nf<-layout(matrix(c(1,2,3,4),ncol=2,byrow=T),
widths=c(rep(10,4)),heights=c(rep(10,4)),respect=T)

f<-outer(x,y,normal.bivariada,rho=0.85,mu1=0,sigma1=1,
mu2=0,sigma2=1)

par(mar=c(1,1,4,1))

persp(x,y,f,theta = 30, phi = 30,  col = "lightblue",
xlab = "X", ylab = "Y", zlab =
"Z",main="rho=0.85",cex.main=0.8)

f<-outer(x,y,normal.bivariada,rho=0.5,mu1=0.5,sigma1=1,
mu2=0,sigma2=1)

par(mar=c(1,1,4,1))

persp(x,y,f,theta = 30, phi = 30,  col = "lightblue",
xlab = "X", ylab = "Y", zlab =
"Z",main="rho=0.5",cex.main=0.8)

f<-outer(x,y,normal.bivariada,rho=0.0,mu1=0,sigma1=1,
mu2=0,sigma2=1)

par(mar=c(1,1,4,1))

persp(x,y,f,theta = 30, phi = 30,  col = "lightblue",
xlab = "X", ylab = "Y", zlab =
"Z",main="rho=0.0",cex.main=0.8)

f<-outer(x,y,normal.bivariada,rho=-0.85,mu1=0,sigma1=1,
```

```
mu2=0,sigma2=1)

par(mar=c(1,1,4,1))

persp(x,y,f,theta = 30, phi = 30,  col = "lightblue",
xlab = "X", ylab = "Y", zlab =
"Z",main="rho=-0.85",cex.main=0.8)

par(oma=c(1,1,1,1),new=T,font=2,cex=1)
mtext(outer=T,"Distribución Normal
Bivariada",side=3,cex=0.8)
```

Modelos DEM: El siguiente ejemplo fue tomado del help de R, con algunas modificaciones:

```
nf<-layout(matrix(c(1,2,3,4),ncol=2,byrow=T),
widths=c(rep(12,4)),heights=c(rep(12,4)),respect=T)

data(volcano) z <- 2 * volcano  # Exaggerate the relief

x <- 10 * (1:nrow(z)) # 10 meter spacing (S to N)


y <- 10 * (1:ncol(z)) # 10 meter spacing (E to W)

par(mar=c(1,1,4,1))
persp(x, y, z, theta = 120, phi = 15, scale = FALSE,
axes = FALSE)

# Now something more complex
# We border the surface, to make it more "slice like"

# and color the top and sides of the surface differently.

zmin <- min(z) - 20
z <- rbind(zmin, cbind(zmin, z, zmin), zmin)
```

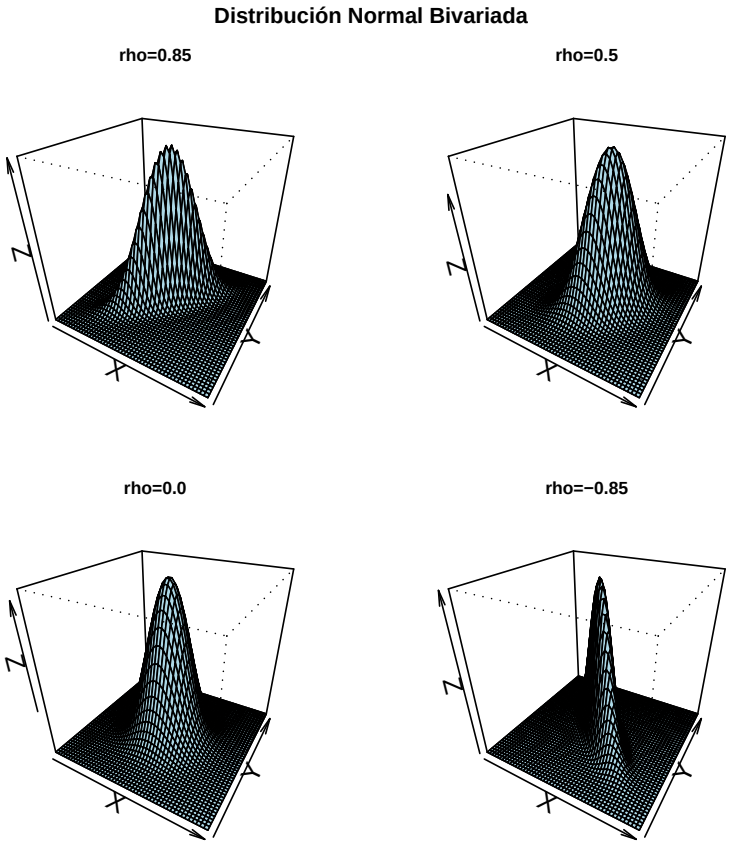

Figura 7.11: *visualización de las superficies normales bivariadas para unos parámetros específicos es posible gracias a la función 'persp'.*

```
x <- c(min(x) - 1e-10, x, max(x) + 1e-10)

y <- c(min(y) - 1e-10, y, max(y) + 1e-10)

fill <- matrix("green3", nr = nrow(z)-1, nc = ncol(z)-1)
fill[,1] <- "gray"
fill[,ncol(fill)] <- "gray"
fill[1,] <- "gray"
fill[nrow(fill),] <- "gray"

par(bg = "lightblue")
par(mar=c(1,1,4,1))

persp(x, y, z, theta = 120, phi = 15, col = fill,
scale = FALSE, axes = FALSE)

par(bg = "slategray")
par(mar=c(1,1,4,1))

persp(x, y, z, theta = 135, phi = 30, col = fill,
scale = FALSE, ltheta = -120, lphi =
15, shade = 0.65, axes = FALSE)

par(mar=c(1,1,4,1))

persp(x, y, z, theta = 135, phi = 30, col = "green3",
scale = FALSE, theta = -120, shade
= 0.75, border = NA, box = FALSE)

par(oma=c(1,1,3,1),new=T,font=2,cex=1)
mtext(outer=T,"Maunga Whau\nUno de 50 Volcanes en
la región de Auckland",side=3,cex=0.9,font=4)
```

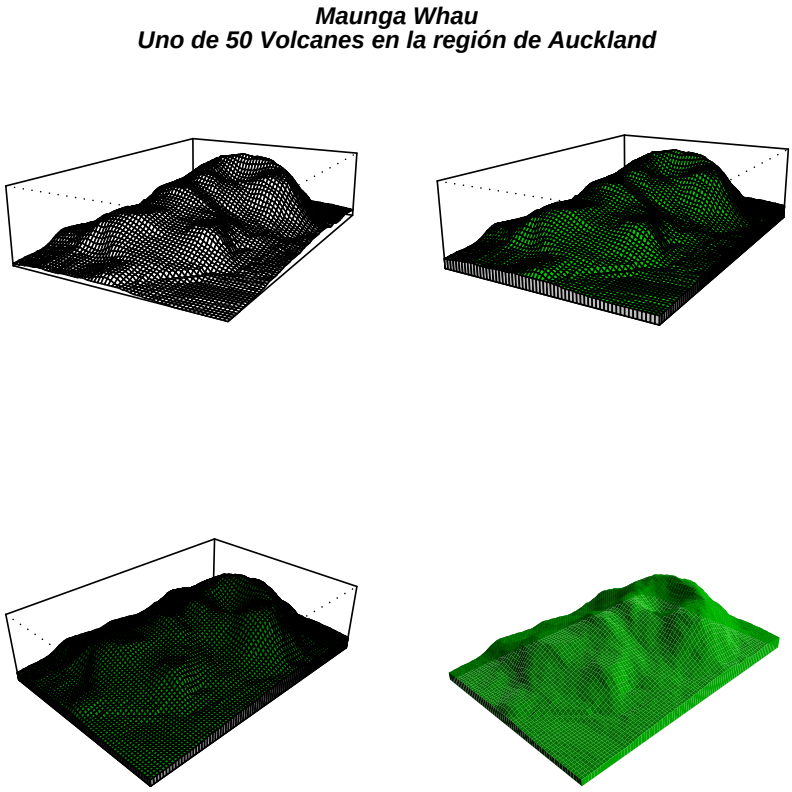

Figura 7.12: *Este gráfico tridimensional es posible en R mediante la función 'persp'*

7.9 Función `mtext`

Esta funció permite escribir texto en una de las cuatro márgenes de la región de la figura o de las márgenes exteriores de la región del dispositivo:

```
mtext(text, side = 3, line = 0, outer = FALSE, at = NULL,
adj = NA, cex = NA, col = NA,
font = NA, vfont = NULL, ...)
```

Argumentos:

- text: Uno o más cadenas de caracteres o expresiones.
- side: Lado del gráfico sobre el cual se ubicará texto (1=abajo, 2=izquierda, 3=arriba, 4=derecha).
- line: Sobre cual linea de margen, empezando en 0 y contando hacia afuera.
- outer: Para usar márgenes exteriores cuando han sido definidas (con par(mar=(..)).
- at: Para dar la ubicación en coordenadas de usuario. Si 'length(at)==0' (el valor por defecto), la ubicación será determinada por 'adj'.
- adj: Ajuste para cada cadena. Para cadenas paralelas a los ejes, 'adj=0' significa alineación izquierda o abajo, y 'adj=1' significa alineación derecha o arriba. 'adj' no es un valor finito (el valor por defecto), el valor 'par("las")' determina el ajuste. Para cadenas graficadas paralelas a los ejes el valor por defecto es centrar la cadena.
- ...: Los parámetros gráficos adicionales son:
  - cex
  - col
  - font
  - vfont

Detalles:

- Las coordenadas de usuario en las márgenes exteriores siempre varían de 0 a 1, no son afectadas por las coordenadas de usuario en la región de la figura.
- Los argumentos 'side', 'line', 'at', 'adj', los demás parámetros gráficos e incluso 'outer' pueden ser vectores, y un reciclamiento ocurrirá para graficar tantas cadenas como el más largo de los argumentos vectoriales. Notar que un vector 'adj' tiene un significado diferente al que tiene en la función 'text'.
- 'adj = 0.5' centrará la cadena, pero para 'outer=TRUE'en la región del dispositivo en vez de la región del gráfico.
- El parámetro 'las' determinará la orientación de las cadenas. Para cadenas graficadas perpendicularmente a los ejes la justificación por defecto es colocar el extremo de la cadena más cerca al eje sobre la linea especificada.
- Notar que si el texto va a ser graficado perpendicular al eje, 'adj' determina la justificación de la cadena y la posición a lo largo del eje a menos que se haya especificado 'at'.

Ver también: 'title', 'text', 'plot', 'par'; 'plotmath' para detalles sobre anotación matemática.

7.10 Función `text`

Esta función, como se ha visto en diferentes ejemplos, permite agregar texto a un gráfico, dibujando las cadenas de caracter dadas en el vector 'labels' en las coordenadas dadas por x y y. y puede ser omitida dado que 'xy.coords(x,y)' es usada para la construcción de las coordenadas.

```
text (x, ...) text.default (x, y = NULL,
labels = seq(along = x), adj = NULL, pos = NULL,
offset = 0.5, vfont = NULL, cex = 1, col = NULL,
font = NULL, xpd = NULL, ...)
```

Argumentos:

- x, y: Vectores numéricos que dan las coordenadas donde el texto dado en 'labels' será escrito. Si la longitud de x e y difieren, el más corto es reciclado.

- labels: Una o más cadenas de caracteres o expresiones que especifican el texto a ser escrito.

- adj: Uno o dos valores en $[0, 1]$ que especifica el ajuste x (y opcionalmente y) de las etiquetas. En la mayoría de los dispositivos valores por fuera de este intervalo también funcionan.

- pos: Un especificador de posicón para el texto. Si es especificado esto hace que se ignore cualquier valor 'adj' dado. Valores de 1, 2, 3, y 4, indican respectivamente las posiciones debajo, a la izquierda, arriba y a la derecha de las coordenadas especificadas.

- offset: Cuando 'pos' es especificado, este valor da el *offset* de la etiqueta desde la coordenada especificada en fracciones de un ancho de caracter.

- vfont: Si un vector caracter de longitud 2 es especificado, entonces fuentes vector Hershey son usadas. El primer elemento del vector selecciona el *typeface* y el segundo selecciona un estilo.

- cex: Factor de expansión de caracter; multiplicado por 'par("cex") produce el tamaño final del caracter.

- col, font: El color y fuente a ser usado; estos por defecto corresponden los valores de los parámetros gráficos globales en 'par()'.

- xpd: Dónde debería ocurrir el *clipping*. Por defecto es 'par("xpd")'.

- ...: Parámetros gráficos adicionales (de 'par').

Detalles:

- Las etiquetas deben ser de tipo caracter o expresión. En el último caso existen disponibles notaciones matemáticas tales como sub y superíndice, letras griegas, fracciones, etc.

- 'adj' permite el ajuste del texto con respecto a (x, y). Valores de 0, 0.5 y 1 especifican izquierda/abajo, medio/medio y derecha/arriba, respectivamente. El valor por defecto es para texto centrado, es decir 'adj=c(0.5,0.5)'. El centramiento vertical exacto necesita información métrica sobre caracteres individuales, lo cual sólo está disponible en algunos dispositivos.

- Los argumentos 'pos' y 'offset' pueden usarse en conjunto con valores retornados por 'identify' para recrear un gráfico etiquetado interactivamente.

- El texto puede ser rotado usando parámetros gráficos 'srt' (ver 'par'); Esto rota alrededor del centro ajustado pot 'adj'.

- Los parámetros 'col', 'cex' y 'font' pueden ser vectores y serán aplicados cíclicamente a los elementos en 'labels'.

Ver también: 'title', 'text', 'plot', 'par'; 'plotmath' para detalles sobre anotación matemática.

Referencias

- Allen, T. y Buckner, G. (1992) Expanding the Role of the Bar Chart in Representing Crime Data. *Chance*, Vol. 5, No. 3-4, pp. 54-59
- Barnett, V. y Lewis, T. (1998) *Outliers in Statistical Data 3rd. ed.* John Wiley & Sons. pp. 308-309
- Benjamini, Y. (1988) Opening the Box of a Boxplot. *The American Statistician*, Vol. 42, No. 4, pp. 257-262
- Burn, D.A. (1993) Designing Effective Statistical Graphs. Publicado por Rao, C.R. en *Handbook of Statistics, Vol. 9.* Elsevier Science Publishers B.V.
- Campbell, S.K. (1990) *Equívocos y Falacias en la Interpretación de Estadísticas.* Editorial Limusa: México.
- Carr, D. y Sun, R. (1999) Using Layering and Perceptual Grouping in Statistical Graphics. *Statistical Computing & Statistical Graphics Newsletter*, Vol. 10, No. 1, pp. 25-31
- Carr, D.B., Somogy, R. y Michaels, G. (1997) Templates for Looking at Gene Expression Clustering. *Statistical Computing & Statistical Graphics Newsletter*, Vol. 7, pp. 20-29
- Carr, D.B. et al. (1998-1999) Boxplot Variations in a Spatial Context: An Omernik Ecoregion and Weather Example. *Statistical Computing & Statistical Graphics Newsletter*, Vol. 9, No.2, pp. 4-15
- Chernoff, H. (1973). The Use of Faces to represent Points in k-Dimensional Space Graphically. *Journal of the American Statistical Association.* Vol. 68, No. 342, pp. 361-367

- Cleveland, W.S. (1985). *The Elements of Graphing Data*. Wadsworth, Inc.:Monterey, CA
- Cleveland, W.S. y McGill, R. (1985). Graphical Perception and Graphical Methods for Analyzing Scientific Data. *Science*. Vol. 229, pp. 828-833
- Costigan-Eaves, P. y Macdonald-Ross, M. (1990) William Playfair (1759-1823). *Statistics Science*, Vol. 5, No. 3, pp. 318-326
- Embrechts, P. y Herzberg, A.M. (1991) Variations of Andrews' Plots. *International Statistical Review*, Vol. 59, No. 2, pp. 175-194
- Fienberg, S.E. (1979). Graphical Methods in Statistics. *The American Statistician*. Vol. 33, No. 4, pp. 165-178
- Flury, B. y Riedwyl, H. (1981) Graphical Representation of Multivariate Data by Means of Asymmetrical Faces. *Journal of the American Statistical Association*. Vol. 76, No. 376, pp. 757-765
- Fisher, N. I. y Switzer, P. (2001). Graphical Assessment of Dependence: Is a Picture Worth 100 Test?. it The American Statistician. Vol. 55, No. 3, pp. 233-239.
- Frigge, M., Hoaglin, D.C. y Iglewicz, B. (1989) Some Implementations of the Boxplot. *The American Statistician*, Vol. 43, No. 1, pp. 50-54.
- Galbraith, R.F. (1989) The Radial Plot: Graphical Assessment of Spread in Ages. *Int. Radiat. Appl. Instrum., Part D*. pp. 207-214
- Hofmann, H. (1998-1999) Simpson on Board the Titanic? Interactive Methods for Dealing with Multivariate Categorical Data. *Statistical Computing & Statistical Graphics Newsletter*, Vol. 9, No. 2, pp. 16-19
- Moore, D.S. (1979) *Statistics: Concepts and Controversies*. W. H. Freeman and Company: San Francisco.
- Scott, D.W. (1979) On Optimal and Data-Based Histograms. *Biometrika*, Vol. 66, No. 3, pp. 605-610
- Tufte, E. (1983) *The Visual Display of Quantitative Information*. Graphics Press: Cheshire

- Tukey, J.W. (1990) Data-Based Graphics: Visual Display in the Decades to Come. *Statistical Science*, Vol. 5, No. 3, pp. 327-339

- Tukey, J.W. (1977) *Exploratory Data Analysis*. Addison-Wesley Publishing Company: Reading, Massachusetts

- Wainer, H. (1981). Graphical Data Analysis. *Ann. Rev. Psychol.* Vol. 32, pp. 191-204

- Wakimoto, K. y Taguri, M. (1978). Constellation Graphical Method for Representing Multi-Dimensional Data. *Annals of the Institute of Statistical Mathematics*. Vol. 30, No. 1, A, pp. 97-104

- Wainer, H. (1984). How to Display Data Badly. *The American Statistician*. Vol. 38, No. 2, pp. 137-147

- Wainer, H. (1990) Graphical Visions from William Playfair to John Tukey. *Statistical Science*, Vol. 5, No. 3, pp. 340-346

- Wegman, E.J., Carr, D.B. y Luo, Q. (1992). Visualizaing Multivariate Data. Technical Report No. 85. Center for Computational Statistics. George Mason University.

- Zani, G., Riani, M. y Corbellini, A. (1998) Robust Bivariate Boxplots and Multiple Outlier Detection. *Computational Statistics & Data Analysis*, Vol. 28, pp. 257-270