Capitulo2

Daniel Villatoro

23/1/2020

Exploratorio vs Gráficos explaneatorios

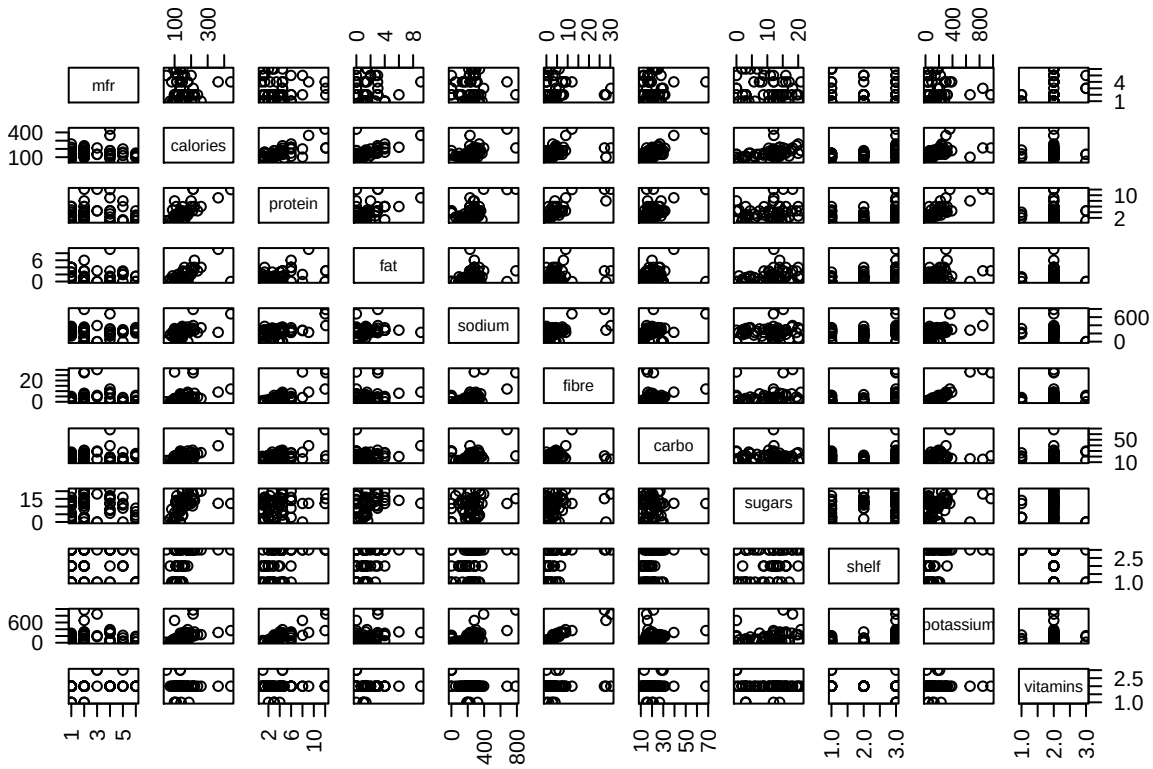
Ambos exploratorio y gráficos exploratorios son relevantes aquí cuando tu necesitas tener un sentido de que es lo que esta dentro de tu data ser traduce esto, en un medio visual que puede ayudar rápidamente a identificar estas características incluyendo curvas lineas tendencias y anomalías.

La segunda observación concierne al nivel de dalles que son apropiados para el análisis exploratorio: Exploración es en general lo mejor hecho en un alto nivel de singularidad donde existe demasiado ruido en los datos pero si tu los sobresimplificas en lineas ecternas con demasiada informacion podrias terminar con la perdida de informacion importante

En contraste el proposito explaneatorio es lo que conlleva a la conclusion del analusta acerca de que hay dentro de los otros datos, consecuentemente es importante las vizualizaciones explaneatorias para remover o minimizar datos detallados, que podria obscurecer el mensaje.

Los siguientes ejemplos muestran las diferencias entre exploratorio y explanatorio vizualizaciones, ambos estan basados en el UScereal data frame from MASS package, el cual describe 65 desayunos a nase de cereales disponiblesen venta en estados unidos, basado en la esta informacion tomada del paquete requerido por la FDA.

```
library(MASS)
plot(UScereal, las = 2)
```



con este comando se muestra un conjunto de arrays Los elementos diagonales de esta matriz enumeran el nombre de la variable que aparece en el eje x de todas las gráficas en esa columna y el eje y de todas las gráficas en esa fila. Como hay 11 variables en este marco de datos, el resultado es una matriz de 110 campos, haciendo que el resultado sea visualmente desalentador a primera vista. Además, porque hay muchos campos incluidos en esta matriz, cada una es tan pequeña que es imposible ver, muchos detalles en cualquier campo individual. Sin embargo, esta matriz representa un útil herramienta para la exploración de datos preliminares porque nos permite escanear rápidamente trazados para ver si parece existir alguna relación fuerte entre alguna de las pares de variables Aquí, parece haber fuertes relaciones entre grasas y calorías (fila 2, columna 4 o viceversa: fila 4, columna 2), entre carbohidratos y calorías (fila 2, columna 7 y viceversa), y entre potasio y fibra (fila 6, columna 10 y viceversa). Además, esta pantalla deja en claro que ciertas variables, por ejemplo, anaquel y vitaminas, exhiben solo unos pocos valores distintos.Si bien esta información se puede obtener utilizando una combinación de otras pantallas, y / o herramientas no gráficas en R, esta matriz de trazado rápida y simple proporciona mucha información preliminar útil si la miramos con suficiente cuidado. Ese dicho, esta matriz de trama no es una buena visualización explicativa porque contienedemasiados detalles extraños para cualquier historia que deseemos contar sobre cualquier par variable individual.

Gráficos en el sistema de R

El sistema mas simple en R es la base grafica y es usada para crear la mayoria de gráficos mostrados.

Funcion Tipo Grafica Generada

plot Many Depende del tipo de objeto barplot Numeric Grafico de Barra boxplot Formula, numeric, or list Caja grafica sumario hist Numeric Histograma sunflowerplot Numeric + Numeric Grafica de girasol mosaicplot Formula or table Grafica de mosaico symbols Multiple numeric Grafico de burbuja

Base grafica

En la anterior lista se enumera algunas de las funciones gráficas básicas más comunes, junto con los tipos de objetos R que pueden aceptar como argumentos de datos de trazado y los tipos de trama que generan. Como

se señaló, la función de trazado básica enumerada allí es genérica, aceptar muchos tipos diferentes de objetos R y generar muchos tipos diferentes de campos como resultado.

Gráficos de cuadricula

El paquete de cuadrícula proporciona un conjunto separado de herramientas básicas.proporciona funciones para dibujar trazados completos, por lo que no se usa con frecuencia directamente para producir gráficos estadísticos. Es más común usar uno de los paquetes de gráficos que se construyen sobre la cuadrícula, especialmente el lattice packages o el ggplot2 package Para construir una pantalla gráfica usando el paquete de cuadrícula, los paso basicos son:

• crear una ventana gráfica; • poner una colección de objetos gráficos en la ventana gráfica; • renderizar la ventana gráfica para obtener una visualización gráfica.

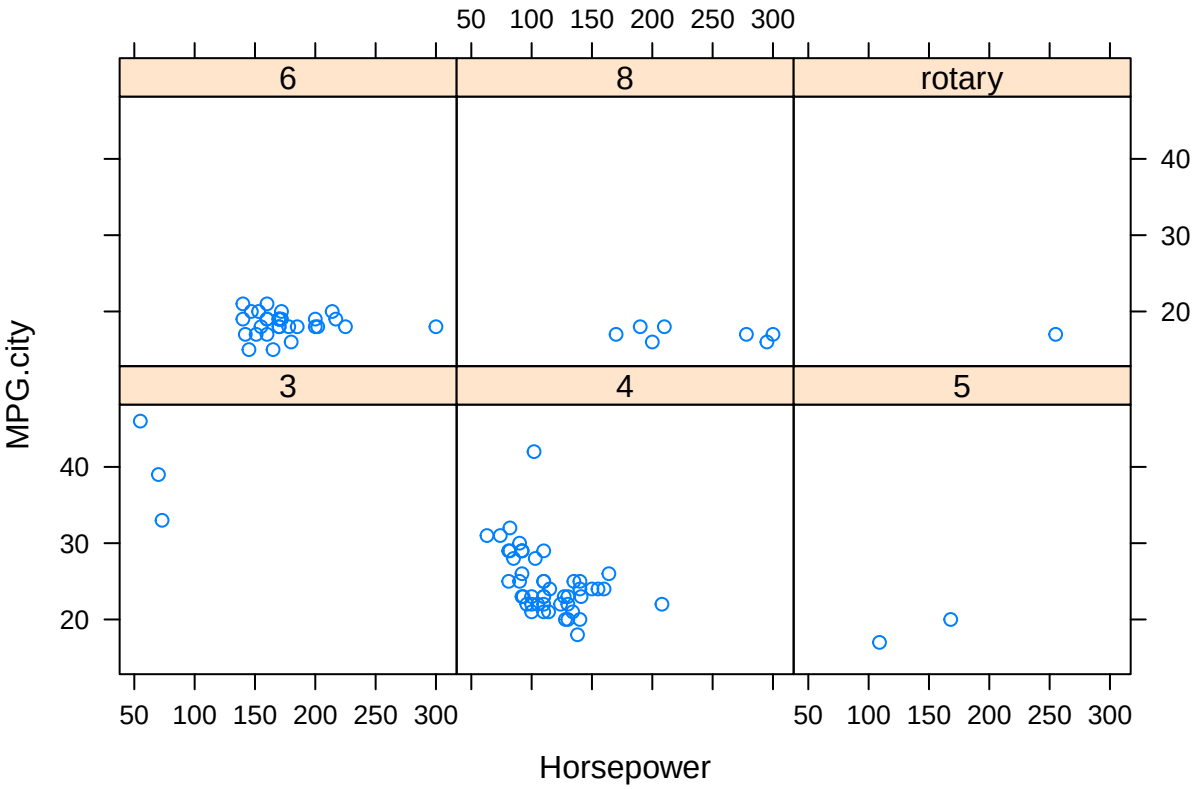
Finalmente, es importante tener en cuenta otros dos puntos. Primero, una fuente potencial de confusión es la existencia de la función de cuadrícula, que es parte de los gráficos base sistema y no relacionado con el paquete de cuadrícula. La cuadrícula de funciones gráficas base agrega una cuadrícula rectangular a una trama gráfica de base existente; referirse a los resultados de ayuda (cuadrícula) para más detalles. El segundo punto importante es que Murrell tiene también desarrolló el paquete gridBase que permite gráficos tanto de cuadrícula como de base para ser usados juntos. Para una introducción a lo que es posible y algunos preliminares.

Lattice Gráficos

proporciona una implementación alternativa de muchas de las funciones de trazado opciones estandar disponibles en gráficos básicos, incluidos diagramas de dispersión, gráficos de barras, diagramas de caja, his-togramas y gráficos QQ. Dos de las principales ventajas de este paquete sobre la base de los gráficos son, en primer lugar, que muchos prefieren las opciones predeterminadas de la red (por ejemplo, colores, puntos, formas, espacios y etiquetas) sobre los valores predeterminados de base correspondientes. Y, en segundo lugar, que los gráficos de red proporcionan implementaciones simples de ciertos características adicionales.

library(lattice)

```
   library(lattice)
xyplot(MPG.city ~ Horsepower | Cylinders, data = Cars93)
```



Una ventaja del sistema de gráficos de lattice es que puede producir tramas extremadamente sofisticadas a partir de expresiones relativamente simples, especialmente con su función de acondicionamiento multipanel. Sin embargo, el costo de esto es que la tarea de agregar anotaciones simples a un diagrama reticular,como agregar líneas o texto adicionales, es más complejo en comparación con el misma tarea en gráficos tradicionales.

ggplot2 package

ggplot2 está basado sobre la gramática de los gráficos, un enfoque sistemático para construir gráficos, muchas de las opciones predeterminadas para los gráficos generados por ggplot2 se basan en investigación, en percepción humana y, por lo tanto, son preferidos por muchos a la base valores predeterminados de gráficos, también como los gráficos de lattice, ggplot2 proporciona mejores soporte para campos de acondicionamiento multipanel, y porque este paquete es altamente extensible y se ha vuelto extremadamente popular, hay muchas extensiones disponibles en forma de otros paquetes R que proporcionan capacidades adicionales significativas más allá del paquete ggplot2 en sí. Al igual que con los gráficos de lattice, sin embargo, un precio pagado por esta flexibilidad adicional es una curva de aprendizaje más pronunciada; otro es la mayor complejidad de generar múltiples matrices de trama, lo que requiere explícitamente trabajando con ventanas gráficas en cuadrícula.

La fubcion plot

Probablemente la función de gráficos base más utilizada es plot, que es una función genérica, lo que significa que la naturaleza de el trazo que genera depende sobre el tipo de objeto R que le pasamos. En el caso de la función de trazado genérico, esto significa que un comando como “plot (x, y)” puede generar un diagrama de dispersión si x e y son numéricos, un resumen de diagrama de caja si x es categórico e y es numérico, o un diagrama de mosaico si ambas variables son factores.

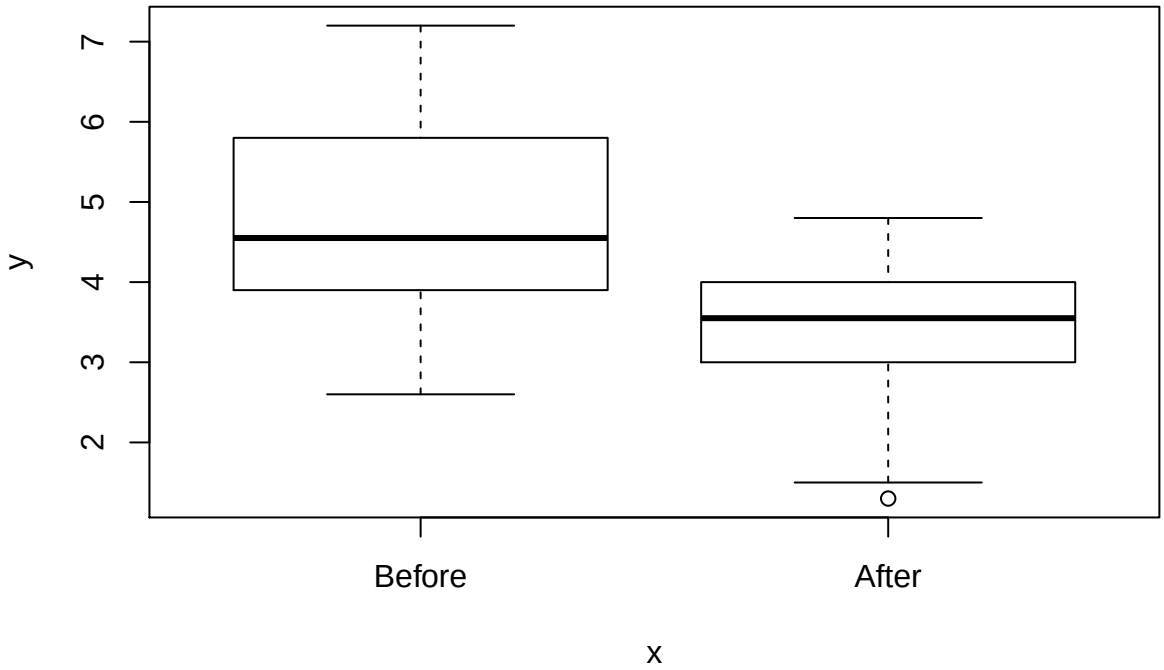
La flexibilidad de la función plot

La flexibilidad de la función de trazado se ilustra en la sesión de muestra R presentado en el Capítulo 1, donde se mostraron los resultados para esta función aplicada a un marco de datos completo, un vector numérico, un factor y un par de variables numéricas: la misma función devolvió una matriz de diagramas de dispersión, una gráfica de la numérica valores en su orden de aparición, un gráfico de barras y un diagrama de dispersión.

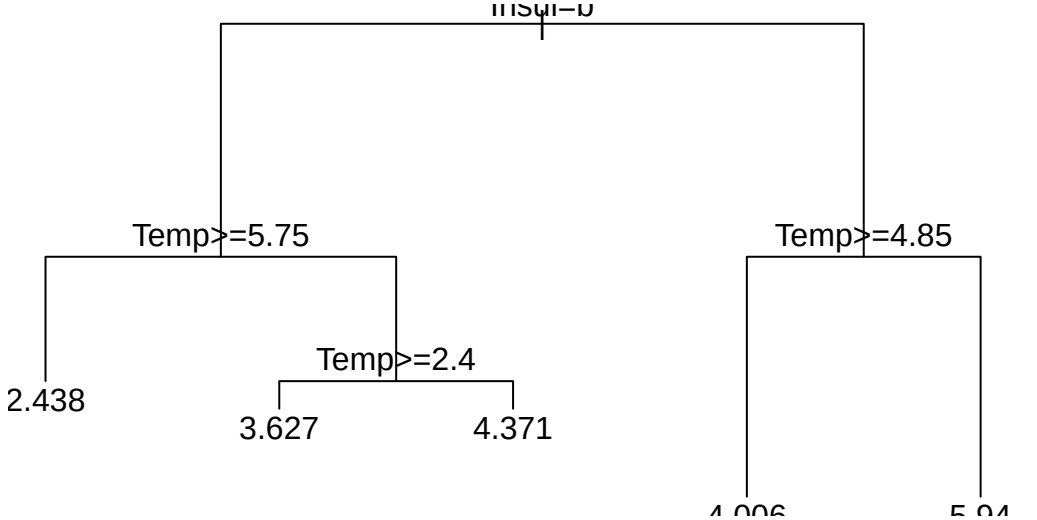
Adicionalmente, esta sesión R de muestra comenzó utilizando la función boxplot para generar un diagrama de caja resumen del consumo de gas de calefacción antes y después de la instalación;

Puede ser generado usando esta funcion:

```
plot(whiteside$Insul, whiteside$Gas)
```




```
library(rpart)
rpartModel <- rpart(Gas ~ ., data = whiteside)
plot(rpartModel)
text(rpartModel)
```



Realmente, la función de trazado solo muestra la estructura de árbol del modelo, sin etiquetas; para obtener las etiquetas, también debemos usar texto, otra función genérica con un Método para objetos rpart: plot(rpartModel) text(rpartModel)

El segundo ejemplo basado en el modelo se muestra en la Fig. 2.5 y pertenece a la clase de modelos MOB, Al igual que el modelo rpart solo descrito, este modelo tiene una estructura basada en árboles, pero en lugar de

generar un solo valor numérico pronosticado para cada nodo terminal del árbol (es decir, cada "Hoja"), cada nodo terminal contiene un modelo de regresión lineal que genera prediciones desde otras covariables, esos modelos podian calzar usando lmtree para partykit package unsando algo similar en el codigo generado usado a partir del modelo rpart.

```
library(partykit)
```

```
## Loading required package: grid
```

```
## Loading required package: libcoin
```

```
## Loading required package: mvtnorm
```

```
MOBmodel <- lmtree(Gas ~ Temp | Insul, data = whiteside)
```

El objeto modelo resultante, MOBmodel, es un objeto S3 de la clase "lmtree" y cuando la función de trazado se aplica a este objeto, obtenemos el resultado que se muestra.

En este caso, el modelo se crea utilizando una estructura de fórmula de tres partes:

#La variable Gas aparece a la izquierda del símbolo ~ para indicar que es la respuesta de la variable a predecir; #La variable Temp aparece entre este símbolo y el | símbolo para indicar que es la covariable utilizada para predecir #La variable de respuesta en los modelos que aparecen en los nodos terminales del árbol #La variableInsul que aparece a la derecha de la | El símbolo es la variable de partición utilizada para construir el árbol, como esta variable de partición es binaria en este ejemplo, el árbol resultante tiene dos nodos, uno correspondiente a los datos "Antes" y el otro correspondiente a los datos "Después".

La estructura de este modelo es clara: todos los registros se asignan a uno de estos nodos, y un registro separado.El modelo de regresión lineal que predice Gas de Temp se construye para cada nodo, en casos favorables, como este, la clase de modelo MOB puede ser extremadamente efectiva en encontrar y explotar una fuerte heterogeneidad en los datos subyacentes.

S3 Clases y funciones genericas

Para cualquier sistema orientado a objetos son los conceptos de clase y método. Una clase define el comportamiento de los objetos describiendo sus atributos y su relación con otras clases. La clase también es utilizado al seleccionar métodos, funciones que se comportan de manera diferente pendiente de la clase de su entrada. Las clases generalmente se organizan en una jerarquía: si no existe un método para un niño, entonces el padre en su lugar se usa el método; el niño hereda el comportamiento del padre.

El nuevo sistema orientado a objetos en R (clases de referencia) tiene esta estructura más tradicional, lo que lo hace muy diferente del sistema S3 Echado a un lado aquí. El sistema S4 también es significativamente diferente del sistema S3: los métodos aún pertenecen a funciones genéricas, pero la estructura es más formal; En los sistemas de gráficos lattice y ggplot2, el sistema S4 tiene una mayor flexibilidad pero una curva de aprendizaje correspondientemente más pronunciada.

Ejemplo

```
class(rpartModel)
```

```
## [1] "rpart"
```

```
text
```

```
## function (x, ...)
## UseMethod("text")
## <bytecode: 0x55b0bb348ed0>
## <environment: namespace:graphics>
```

Parámetros opcionales para gráficos base

Se observó anteriormente que hay 72 parámetros gráficos básicos opcionales que afecta a muchas de las funciones de trazado de gráficos base. Estos parámetros son establecidos por el función par, que también

se puede llamar para devolver una lista con el nombre actual valores para estos parámetros. Los nombres son:Parámetros opcionales para gráficos base

Se observó anteriormente que hay 72 parámetros gráficos básicos opcionales que afecta a muchas de las funciones de trazado de gráficos base. Estos parámetros son establecidos por el función par, que también se puede llamar para devolver una lista con el nombre actual valores para estos parámetros. Los nombres son:

Parámetros opcionales para gráficos base

Se observó anteriormente que hay 72 parámetros gráficos básicos opcionales que afecta a muchas de las funciones de trazado de gráficos base. Estos parámetros son establecidos por el función par, que también se puede llamar para devolver una lista con el nombre actual valores para estos parámetros. Los nombres son:

```
names(par())
```

```
##  [1] "xlog"      "ylog"      "adj"       "ann"       "ask"       "bg"
##  [7] "bty"       "cex"       "cex.axis"  "cex.lab"   "cex.main"  "cex.sub"
## [13] "cin"       "col"       "col.axis"  "col.lab"   "col.main"  "col.sub"
## [19] "cra"       "crt"       "csi"       "cxy"       "din"       "err"
## [25] "family"    "fg"        "fig"       "fin"       "font"      "font.axis"
## [31] "font.lab"  "font.main" "font.sub"  "lab"       "las"       "lend"
## [37] "lheight"   "ljoin"     "lmitre"    "lty"       "lwd"       "mai"
## [43] "mar"       "mex"       "mfcol"     "mfg"       "mfrow"     "mgp"
## [49] "mkh"       "new"       "oma"       "omd"       "omi"       "page"
## [55] "pch"       "pin"       "plt"       "ps"        "pty"       "smo"
## [61] "srt"       "tck"       "tcl"       "usr"       "xaxp"      "xaxs"
## [67] "xaxt"      "xpd"       "yaxp"      "yaxs"      "yaxt"      "ylbias"
```

Varios de estos parámetros vienen en grupos estrechamente relacionados. Uno es el "cex-familia"que especifica en qué medida se deben ampliar el texto y los símbolos en relación con su tamaño predeterminado. Estos parámetros incluyen:

• cex especifica los valores de texto y símbolos de trazado en el siguiente diagrama generado, que sirve como base para todos los demás parámetros en este grupo; • cex.axis especifica la escala de las anotaciones del eje, en relación con cex; • cex.lab especifica la escala de las etiquetas del eje, en relación con cex

La funcion plot

• cex.main especifica la escala del título de la trama principal, en relación con cex; • cex.sub especifica la escala del subtítulo de la trama, en relación con cex.

Es importante tener en cuenta que algunas funciones, como los puntos y el texto, permiten el cex parámetro que se especificará en la llamada de función como un vector, lo que permite puntos en la trama para tener diferentes tamaños. Otras dos familias de parámetros estructuradas de manera similar son la "familia col" que especifica colores para puntos, líneas y texto. Esta familia especifica las opciones de fuente para el texto. apareciendo en gráficos, en notaciones de ejes, en etiquetas y en títulos: • font especifica uno de los siguientes cuatro tipos de fuente para el texto en el diagrama: - font = 1 especifica texto plano; - font = 2 especifica la negrita; - font = 3 especifica cursiva; - font = 4 especifica negrita cursiva. • font.axis especifica la fuente que se utilizará para las anotaciones de eje; • font.lab especifica la fuente que se utilizará para las etiquetas de eje; • font.main especifica la fuente que se utilizará para el título de la trama principal; • font.sub especifica la fuente que se utilizará para el subtítulo de la trama.

El parámetro ask es un indicador lógico (el valor predeterminado es FALSE) que especifica si el sistema de gráficos debe detener y solicitar al usuario una respuesta antes de mostrar el siguiente grafico. Esta opción es útil si queremos ver una secuencia de gráficos individuales, por ejemplo, si creamos un bucle que genera un número de diagramas de dispersión entre diferentes variables y queremos tener tiempo para mira cada uno antes de pasar al siguiente. Esta opción se establece comúnmente por los ejemplos de R integrados que muestran muchas campos, y es importante ser consciente de este uso porque la opción de preguntar no

siempre se restablece correctamente después de estos. Para restablecer el sistema de gráficos para mostrar graficar resultados inmediatamente, entonces es necesario ejecutar el siguiente código:

```
par(ask = FALSE)
```

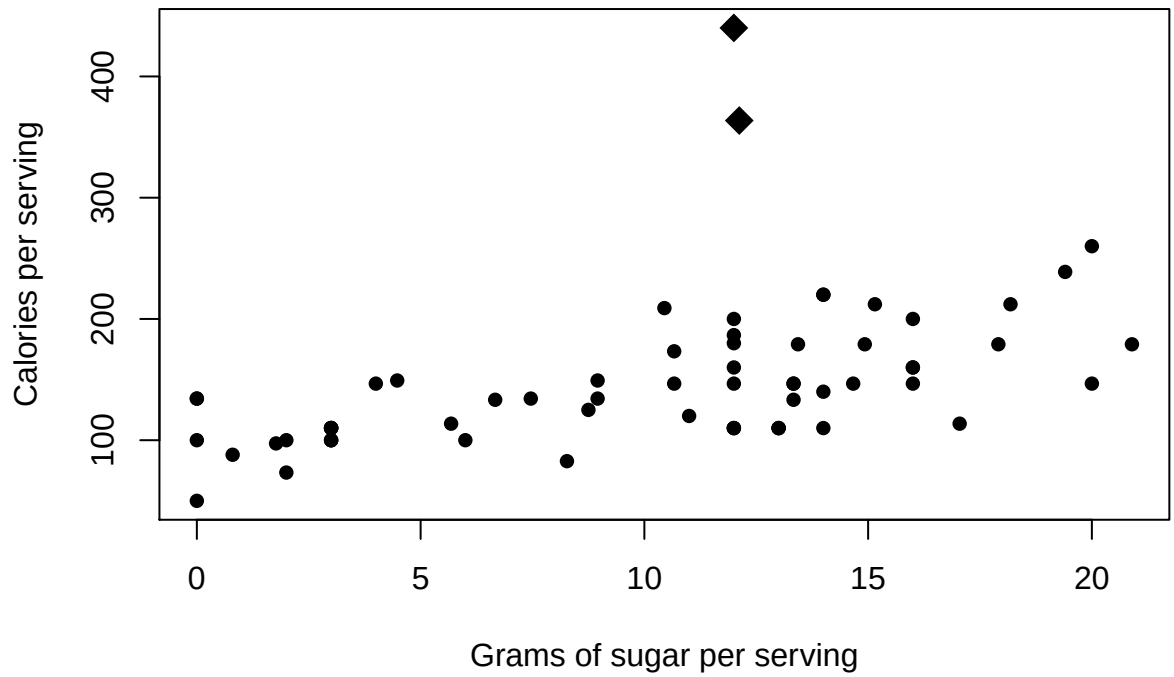
Usando solo sus opciones predeterminadas, las funciones básicas de trazado en R generalmente proporcionan pantallas útiles, especialmente para fines de análisis exploratorio, pero generalmente carecen de detalles importantes que podamos querer agregar, particularmente si planeamos compartir los resultados con otros. Es posible mejorar mucho usando los gráficos base

```
par(mfrow=c(1,1))
x <- UScereal$sugars
y <- UScereal$calories
```

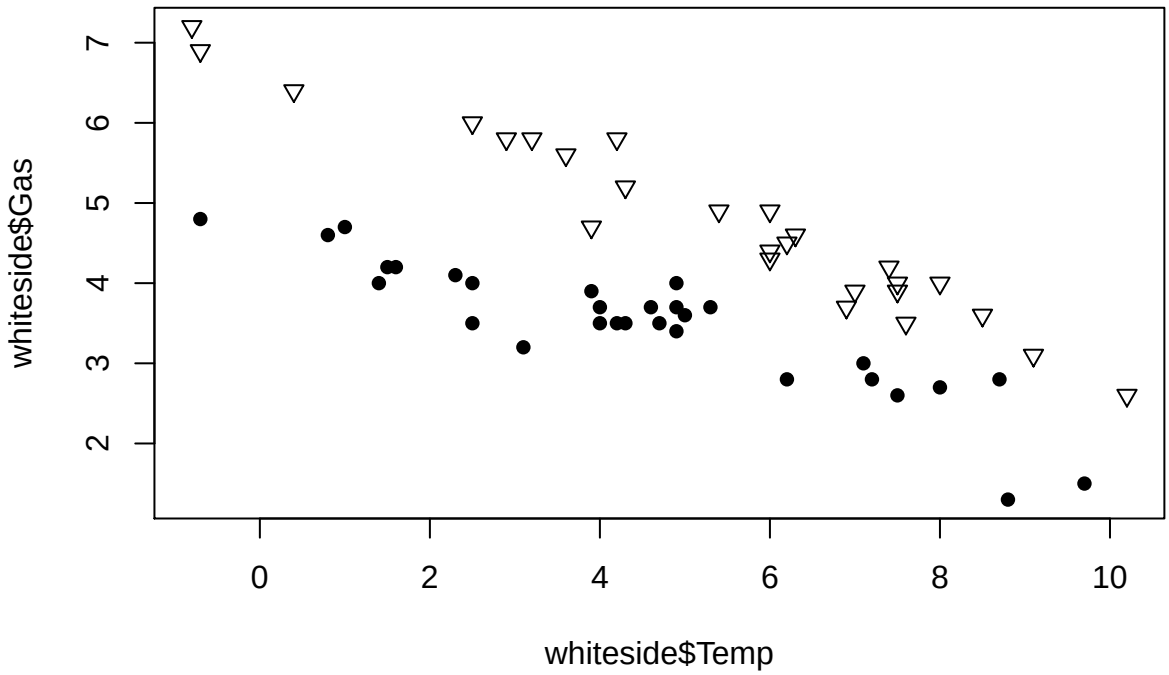
La siguiente línea invoca la función de trazado, especificando tres argumentos opcionales: • xlab es una cadena de caracteres que especifica el texto para la etiqueta del eje x; • ylab es una cadena de caracteres que especifica el texto para la etiqueta del eje y; • type = "n" especifica que el diagrama básico se construye, pero no se muestra. Esta última opción es útil en casos como este donde deseamos mostrar diferentes subconjuntos de datos utilizando diferentes tamaños de puntos o formas.

Específicamente, llamando a La función plot con type = "n" configura el marco básico para el gráfico, incluyendo límites de los ejes X e Y, nombres de etiquetas y cualquier otra opción especificada en el comando de la trama. Aquí, el código es:

```
plot(x, y, xlab = "Grams of sugar per serving",
ylab = "Calories per serving", type = "n")
index <- which(y > 300)
points(x[-index], y[-index], pch = 16)
points(x[index], y[index], pch = 18, cex = 2)
```




```
plot(whiteside$Temp, whiteside$Gas, pch=c(6,16)[whiteside$Insul])
```



Agregando Texto

En el mismo gráfico que en la Fig. 2.7, excepto que la función de texto se ha utilizado para agregar etiquetas de identificación a los dos valores atípicos. Al igual que las funciones de puntos y líneas, el texto agrega detalles al diagrama actual.

Aunque esta función admite otras opciones (consulte la ayuda (texto) para más detalles), las tres básicas son: (1) la posición x del texto, (2) la posición y del texto, y (3) el texto a se visualizará.

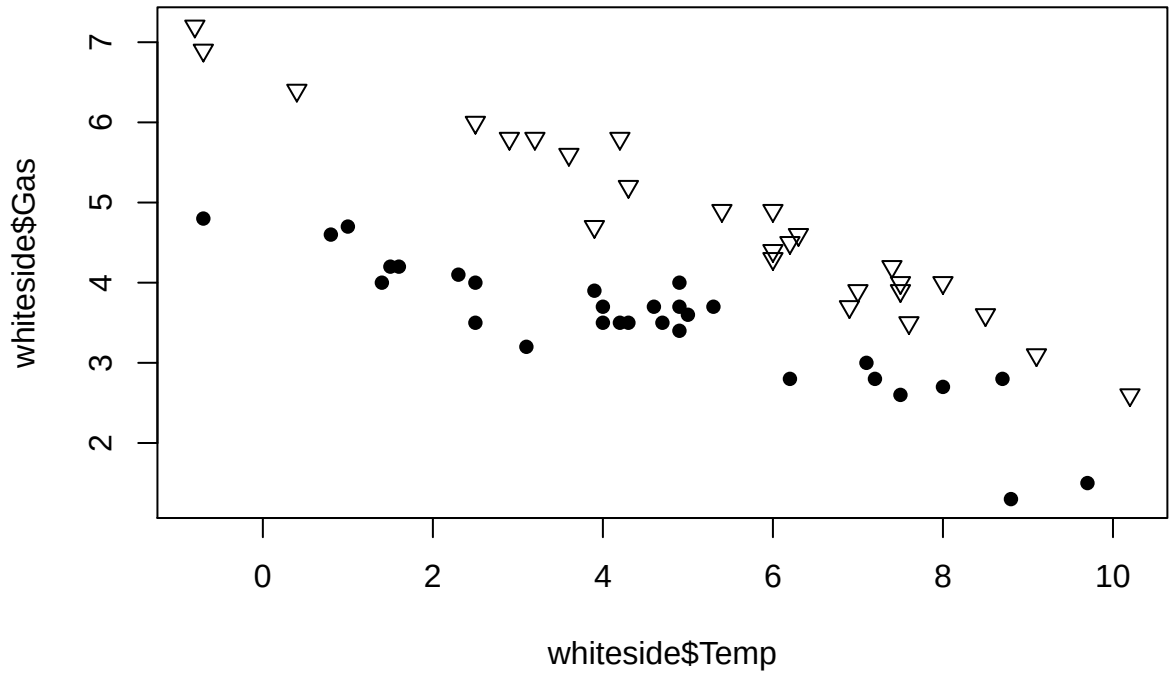
La alineación de izquierda a derecha del texto está determinada por el parámetro adj: por defecto, este parámetro tiene el valor 0.5, lo que hace que el texto estar centrado En este caso, la posición x en la llamada a la función de texto especifica la ubicación del centro del texto; Las alternativas más útiles son adj = 0, que hace que el texto se justifique a la izquierda (es decir, la posición x especificada en la llamada a la función define el extremo izquierdo de la cadena de texto en el diagrama), y adj = 1, que causa el texto que se justificará a la derecha (es decir, la posición x especificada en la llamada a la función define el extremo derecho de la cadena de texto).

Como adorno final de los ejemplos anteriores, la figura 2.9 agrega una leyenda que nos dice algo sobre las dos líneas que aparecen en la trama.

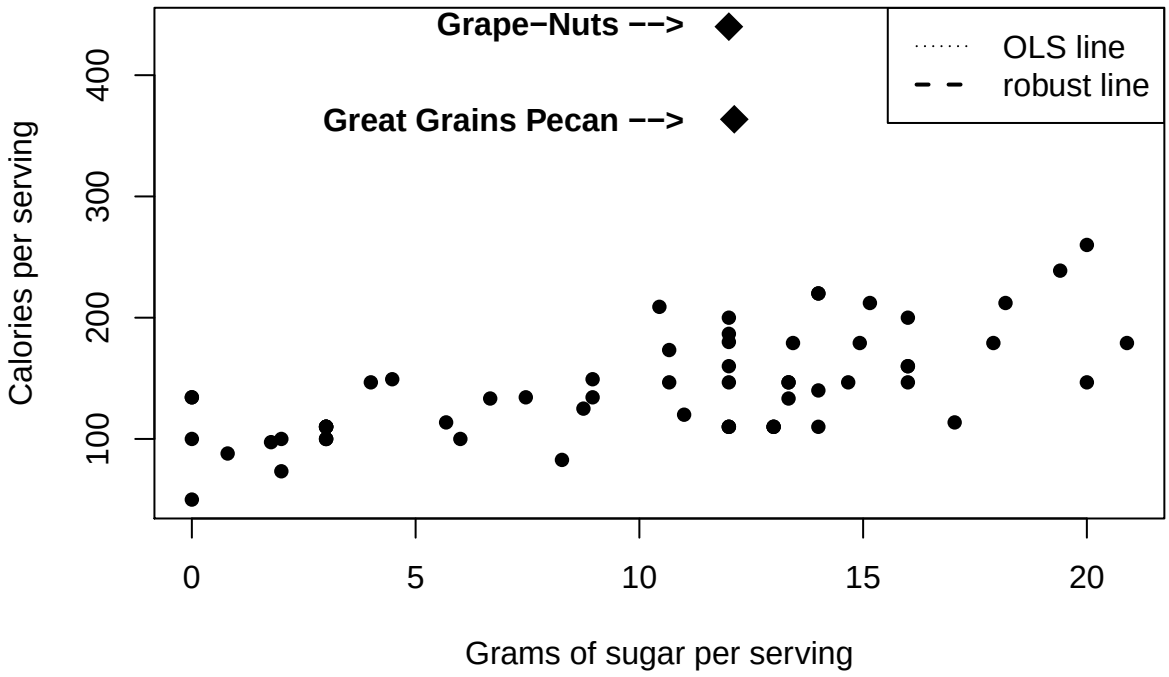
establecido con la función de leyenda, que coloca una visualización de texto explicativo en recuadro en una ubicación especificada en el diagrama actual. En su forma más simple, esta función se puede usar como la función de texto, especificando las posiciones x e y y la cadena de texto que se agregará, pero la función de leyenda tiene muchos más parámetros opcionales, lo que permite una flexibilidad considerable en lo que aparece en el cuadro de leyenda .

El código utilizado para generar este gráfico es idéntico al utilizado para generar el gráfico anterior, excepto que estas dos líneas se agregaron al final:

```
plot(whiteside$Temp, whiteside$Gas, pch=c(6,16)[whiteside$Insul])
```




```
plot(x, y, xlab = "Grams of sugar per serving",
ylab = "Calories per serving", type = "n")
index <- which(y > 300)
points(x[-index], y[-index], pch = 16)
points(x[index], y[index], pch = 18, cex = 2)
pointLabels <- paste(rownames(UScereal)[index], "-->")
text(11, y[index], pointLabels, adj = 1, font = 2)
legend(x = "topright", lty = c(3,2), lwd = c(1,2),
legend = c("OLS line", "robust line"))
```



Personalizando Ejes

Se dispone de una flexibilidad considerable en los gráficos de base R para personalizar los ejes. Ya hemos visto cómo las etiquetas de eje se especifican fácilmente a través de parametros xlab y ylab y cómo se pueden establecer los límites de los ejes x e y utilizando los parámetros xlim e ylim.

Algunos de los otros parámetros gráficos básicos que se pueden configurar mediante la función par también se pueden utilizar para personalizar los ejes en un gráfico.

Uno es el parámetro las, utilizado para especificar la orientación de las etiquetas de los ejes. Este parámetro acepta estos valores enteros:

• las = 0: las etiquetas se muestran paralelas a los ejes (opción predeterminada); • las = 1: las etiquetas son siempre horizontales; • las = 2: las etiquetas son siempre perpendiculares a los ejes; • las = 3: las etiquetas son siempre verticales.

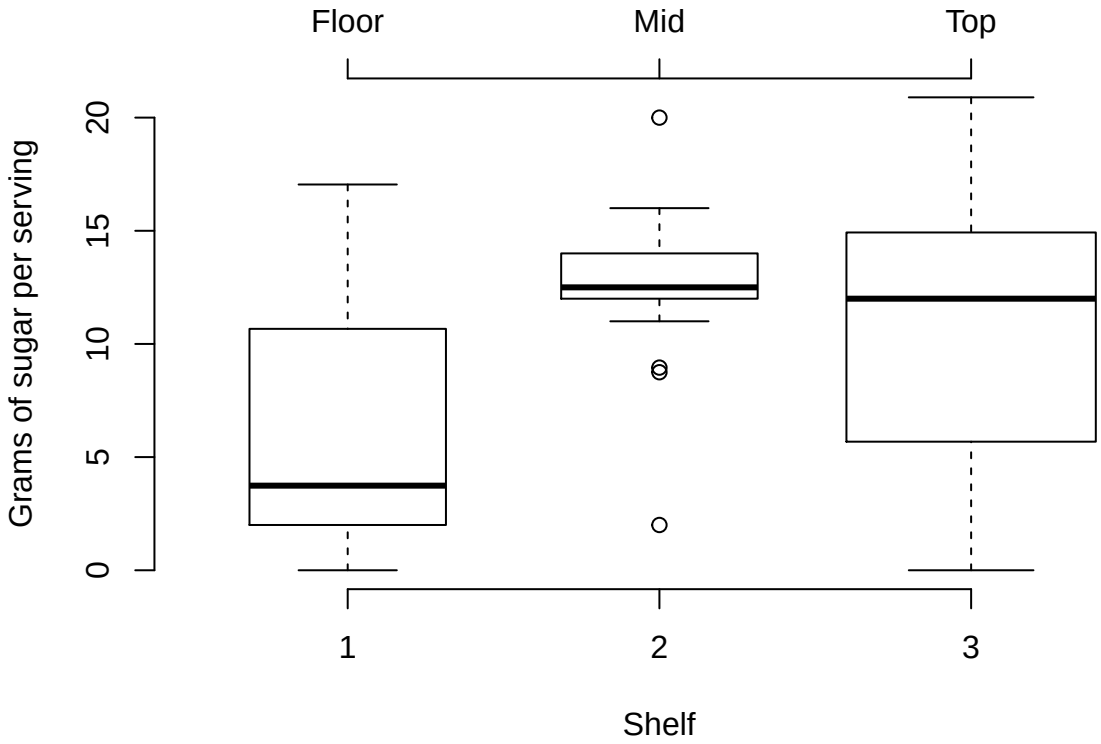
Pasos

1. Ejecute la función de gráficos base deseada con ejes = FALSE;
2. Use la función de eje para especificar sus propios ejes.

Especificar ejes = FALSO suprime los ejes predeterminados normalmente generados con el diagrama y la función de eje le permiten agregar cualquiera de los siguientes cuatro ejes:

• side = 1 crea el eje x inferior (predeterminado), debajo del gráfico. • side = 2 crea el eje y izquierdo (predeterminado), a la izquierda del gráfico. • lado = 3 crea un eje x superior, por encima de la gráfica. • lado = 4 crea un eje y derecho, a la derecha del gráfico.

```
boxplot(sugars ~ shelf, data = UScereal, axes = FALSE,
xlab = "Shelf", ylab = "Grams of sugar per serving",
varwidth = TRUE)
axis(side = 1, at = c(1, 2, 3), labels = c(1, 2, 3))
yRange <- seq(0, max(UScereal$sugars), 5)
axis(side = 2, at = yRange, labels = yRange)
axis(side = 3, at = c(1, 2, 3), labels = c("Floor", "Mid", "Top"))
```



La primera línea construye la visualización básica del diagrama de caja, en este caso, un diagrama de caja de ancho variable, de modo que el ancho de cada diagrama de caja individual refleje el número de cereales diferentes en cada estante. Tenga en cuenta que los parámetros xlab e ylab definen las etiquetas de los ejes x e y, pero como hemos especificado ejes = FALSO, los ejes en sí no se dibujan, solo sus etiquetas. La segunda línea invoca la función de eje para crear el eje x inferior, que coloca las marcas de graduación en los valores numéricos 1, 2 y 3, correspondientes a los tres niveles de la variable de estante, y etiqueta las marcas de graduación con estos mismos números.

Las siguientes dos líneas construyen el eje y a la izquierda de la gráfica: yRange es una secuencia de valores de 0 a valor máximo de azúcares del marco de datos de UScereal en pasos de tamaño 5, que se utiliza para los parámetros at y de las etiquetas para la llamada a la función del eje con lado = 2 para construir el eje y.

Finalmente, la última línea construye un eje x superior sobre el gráfico, con marcas de verificación en la misma ubicación que las del eje x inferior, pero con diferentes etiquetas, dando las posiciones reales.

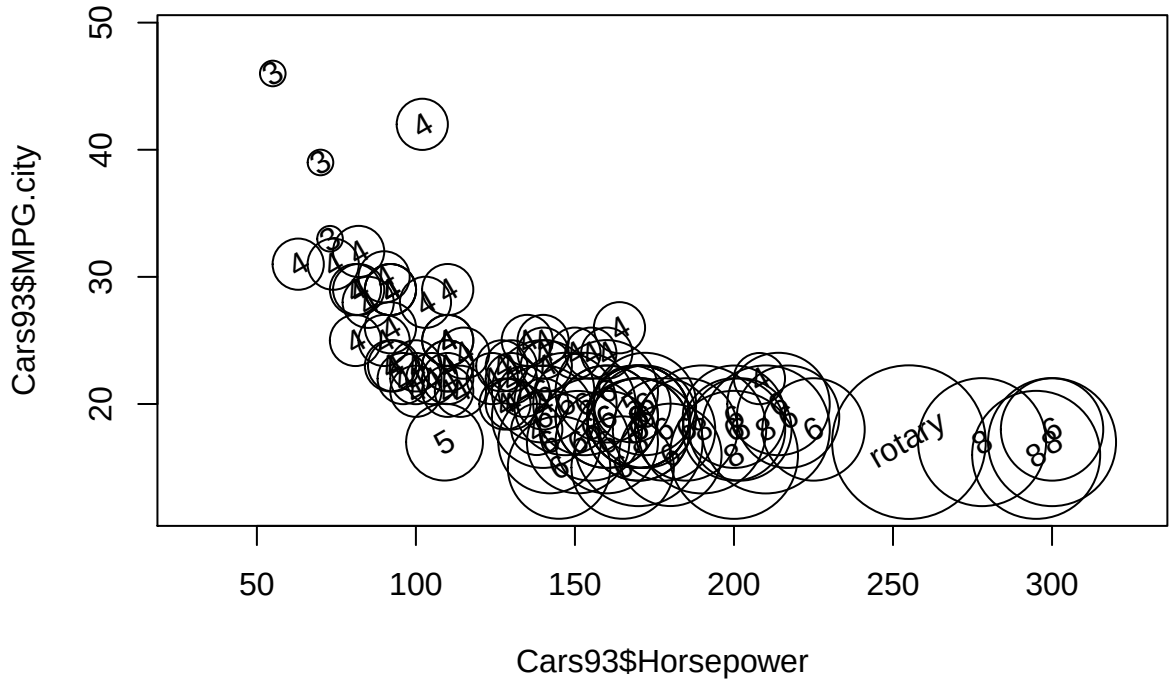
Funcion Simbolo

La función de símbolos es una función gráfica de base extremadamente flexible que admite puertos de la generación de trazos que muestran la relación entre más de dos variables numéricas Un ejemplo específico es el diagrama de burbujas, un diagrama de dispersión de dos variables numéricas, con puntos representados por círculos donde el tamaño de cada círculo está determinado por una tercera variable numérica. Estas parcelas se pueden generar utilizando los símbolos funcionan especificando los argumentos habituales de diagrama de dispersión x e y, junto con un tercer parámetro opcional círculos como un vector numérico z, de la misma longitud que x e y.

Otras opciones disponibles en la función de símbolos incluyen cuadrados, rectángulos, estrellas, termómetros y diagramas de caja.

```
library(MASS)
library(carData)
symbols(Cars93$Horsepower, Cars93$MPG.city,
circles = as.numeric(Cars93$Cylinders),
inches = 0.4)
```

```
text(Cars93$Horsepower, Cars93$MPG.city,
Cars93$Cylinders, srt = 30)
```



Configurando un simple array con mfrow

El parámetro de trazado mfrow es un vector bidimensional que define el número de elementos en una matriz rectangular de gráficos, el primer elemento de este vector especifica el número de filas en esta matriz de trazado, mientras que el segundo elemento especifica el número de columnas. Para crear una matriz con R filas y columnas C, primero especificamos par (mfrow = c (R, C)) y luego usamos cualquiera de las funciones gráficas básicas para crear cada gráfico individual.

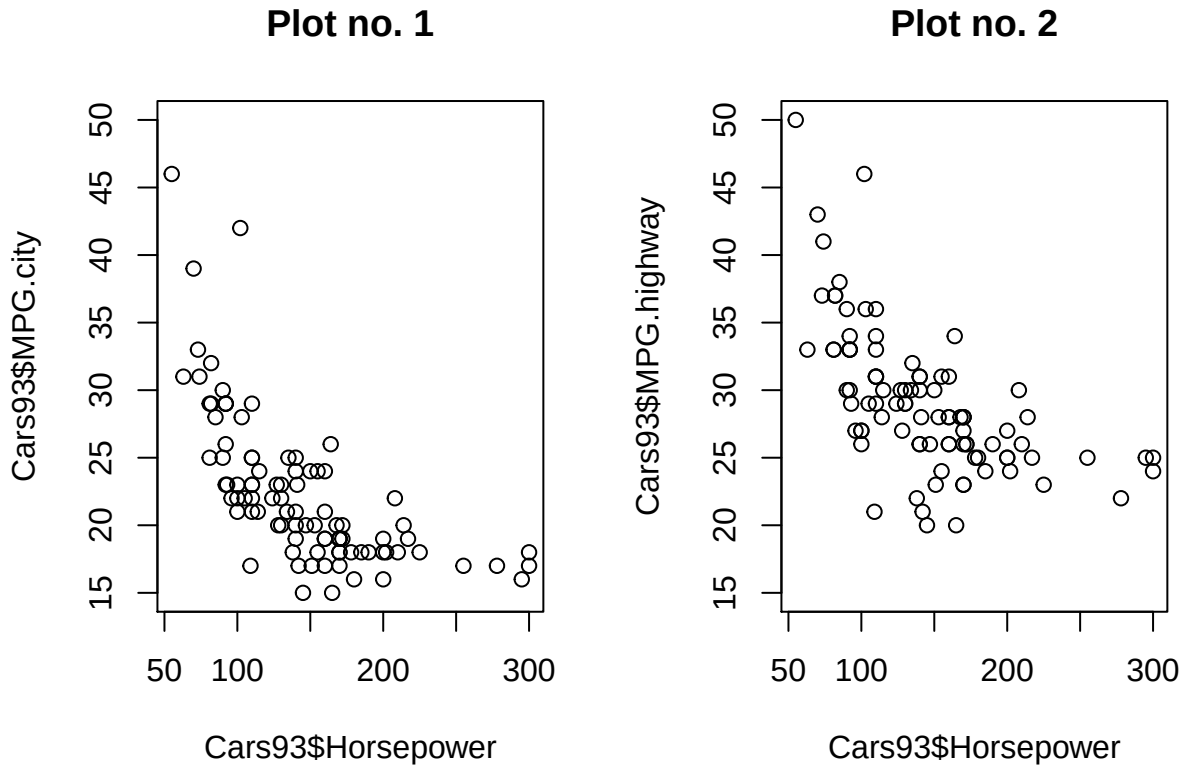
Al crear estos gráficos, procedemos desde la esquina superior izquierda de la matriz, creando los gráficos en la fila superior de izquierda a derecha, luego creamos la segunda fila de izquierda a derecha, repitiendo hasta llegar a la fila inferior, de izquierda a derecha Correcto. En principio, R y C pueden ser enteros positivos, pero si los hacemos demasiado grandes, las parcelas individuales se vuelven muy pequeñas y difíciles de leer.

Esta estructura es extremadamente útil cuando deseamos comparar dos parcelas similares, como en este ejemplo: la gráfica de la izquierda en esta matriz muestra el millaje de gasolina de la ciudad (MPG.city) del marco de datos Cars93, trazada contra la variable Horsepower. Como vimos en el ejemplo anterior, hay una tendencia decreciente más o menos monótona en estos datos: cuanto mayor es la potencia, en general, menor es el consumo de combustible. La gráfica de la derecha muestra la gráfica correspondiente para el millaje de gasolina en la carretera (MPG.highway) contra los caballos de fuerza.

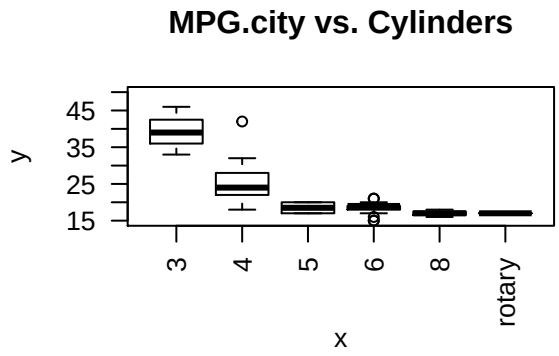
Se ve que estas variables exhiben una relación generalmente similar, aunque la curva de kilometraje de la carretera parece desplazada hacia arriba en relación con la curva de kilometraje de la ciudad, lo que refleja el hecho de que el kilometraje de la carretera es generalmente mayor. Tenga en cuenta que para ver esta diferencia claramente, es necesario especificar los mismos rangos para ambas parcelas; Dado que la variable x es idéntica en estas gráficas, esto significa que necesitamos especificar los mismos límites del eje y para ambas gráficas usando el parámetro ylim. El código R utilizado para generar estos gráficos es:

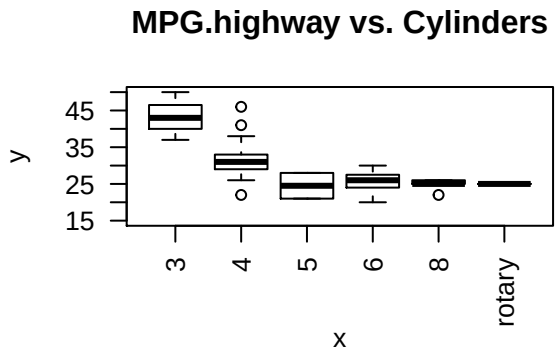
```
library(MASS)
library(carData)
par(mfrow=c(1,2))
plot(Cars93$Horsepower, Cars93$MPG.city, ylim = c(15, 50))
```

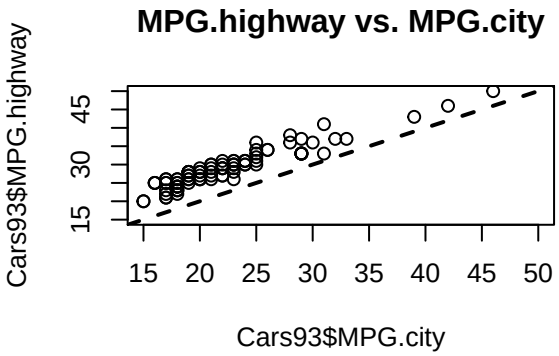
```
title("Plot no. 1")
plot(Cars93$Horsepower, Cars93$MPG.highway, ylim = c(15, 50))
title("Plot no. 2")
```

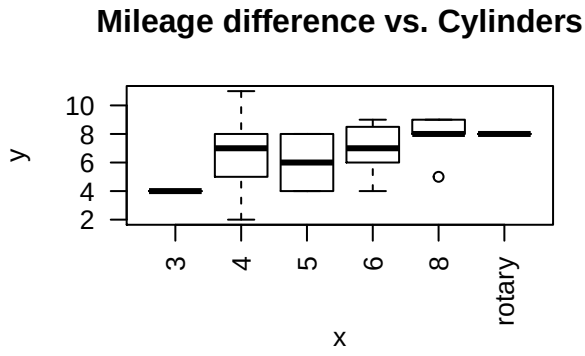



```
par(mfrow=c(2,2))
plot(Cars93$Cylinders, Cars93$MPG.city, las = 2, ylim = c(15, 50))
title("MPG.city vs. Cylinders")
plot(Cars93$Cylinders, Cars93$MPG.highway, las = 2, ylim = c(15, 50))
title("MPG.highway vs. Cylinders")
plot(Cars93$MPG.city, Cars93$MPG.highway, xlim = c(15, 50),
ylim = c(15, 50))
title("MPG.highway vs. MPG.city")
abline(a = 0, b = 1, lty = 2, lwd = 2)
delta <- Cars93$MPG.highway - Cars93$MPG.city
plot(Cars93$Cylinders, delta, las = 2)
title("Mileage difference vs. Cylinders")
```









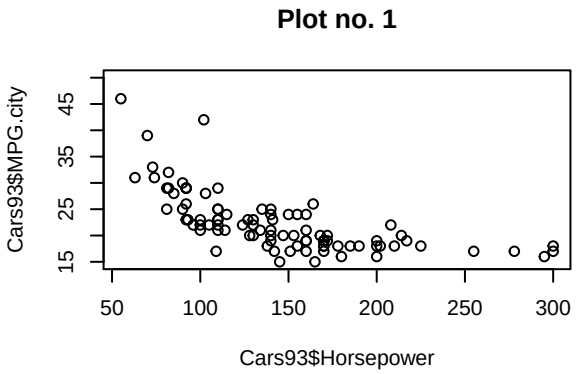
Usando la funcion Layout La función de diseño proporciona un enfoque más flexible para crear matrices de trazado.

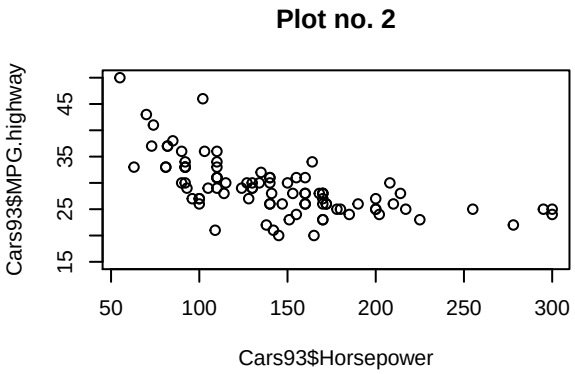
Aunque el uso del diseño es un poco más complicado que el uso del parámetro mfrow, permite una mayor flexibilidad para especificar los tamaños, formas y posiciones de las graficas.

La base de esta flexibilidad es una matriz de diseño, que tiene elementos enteros que especifican las posiciones de la trama, y construir esta matriz correctamente es probablemente el aspecto más desafiante del uso de la función de diseño. Para ver cómo funciona esto, considere el siguiente ejemplo simple:

```
vectorOfNumbers <- c(rep(0, 4), rep(c(1,1,2,2), 2), rep(0, 4))
layoutMatrix <- matrix(vectorOfNumbers, nrow = 4, byrow = TRUE)
```

```
layout(layoutMatrix)
plot(Cars93$Horsepower, Cars93$MPG.city, ylim = c(15, 50))
title("Plot no. 1")
plot(Cars93$Horsepower, Cars93$MPG.highway, ylim = c(15, 50))
title("Plot no. 2")
```





```
layoutVector <- c(rep(c(1,1,0,0,0,0),2), rep(c(0,0,2,2,2,2),4))
layoutMatrix <- matrix(layoutVector, nrow = 6, byrow = TRUE)
layoutMatrix
```

```
##      [,1] [,2] [,3] [,4] [,5] [,6]
## [1,]    1    1    0    0    0    0
## [2,]    1    1    0    0    0    0
## [3,]    0    0    2    2    2    2
## [4,]    0    0    2    2    2    2
## [5,]    0    0    2    2    2    2
## [6,]    0    0    2    2    2    2
```

Colores

Si se usa bien, el color puede agregar mucho a una visualización gráfica de datos, ayudándonos a ver detalles importantes, pero mal utilizados, el color puede ocultar estos detalles por completo.

Para hacer un uso efectivo de los diferentes aspectos del color, Yau propone tres tipos diferentes de escala de colores.

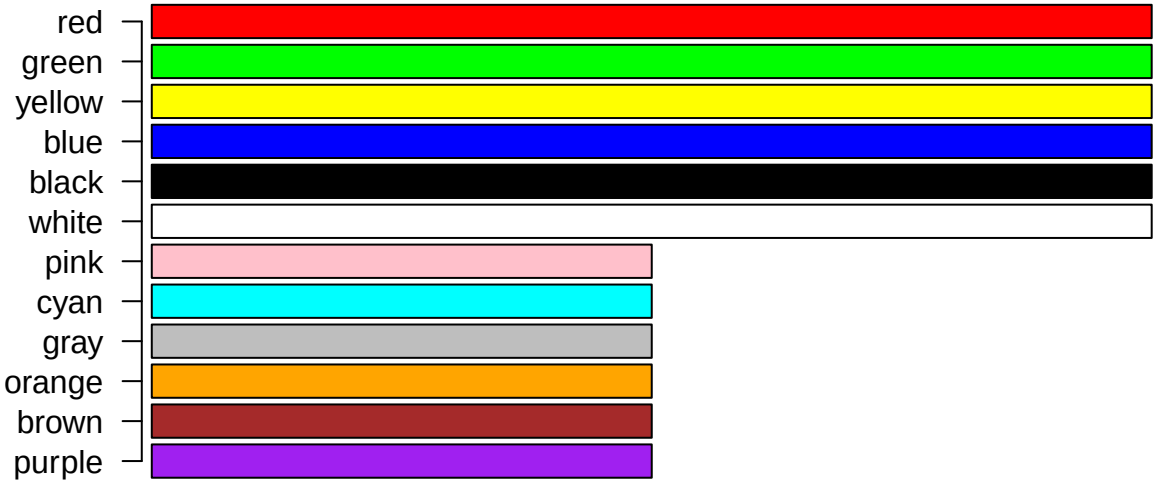
• secuencial: se utilizan niveles de saturación crecientes del mismo tono para representar valores crecientes de una variable no negativa:

divergentes: se usan dos tonos contrastantes, uno para valores negativos y el otro para valores positivos, con niveles de saturación crecientes para indicar "más negativo" o "más positivo", con un tercer color neutro utilizado para indicar cero;

cualitativo: los tonos contrastantes se utilizan para representar los diferentes niveles de una variable categórica no ordenada.

El ejemplo de Tufte descrito anteriormente muestra que también es posible construir escalas de color secuenciales o divergentes basadas en el brillo (es decir, luminancia o tinte) en lugar de saturación (es decir, intensidad), ya que ambos atributos de color están ordenados naturalmente.

```
Top12Colors <- c("red", "green", "yellow", "blue", "black", "white",
"pink", "cyan", "gray", "orange", "brown", "purple")
colorVector <- rev(Top12Colors)
barLengths <- c(rep(1,6), rep(2,6))
yvec <- barplot(barLengths, col = colorVector, horiz = TRUE, axes = FALSE)
axis(2, yvec, colorVector, las = 2)
```



Opciones de Color en R

El sistema de gráficos base en R admite 657 colores, especificados con los parámetros gráficos apropiados.

Específicamente, la familia de parámetros col está estructurada:

• col especifica el color del texto y los símbolos de trazado; • col.axis especifica el color de los ejes; • col.lab especifica el color de las etiquetas del eje; • col.main especifica el color del título de la trama principal; • col.sub especifica el color del subtítulo de la trama.

Un aspecto importante del color es que se puede especificar de varias maneras diferentes, probablemente la más simple es usar una cadena de caracteres como las que se usan para especificar colores de las barras.

El código R utilizado para generar este gráfico fue:

```
set.seed(3)
colorNames <- sort(sample(colors(), size = 50))
plot(seq(1, 10, 1), rep(5, 10), ylim = c(0, 6), xlab = "", ylab = "",
axes = FALSE, type = "n")
for (i in 1:5){
index <- seq(1, 10, 1) + (i-1) * 10
angle <- (-1)^(i+1) * 45
text(seq(1, 10, 1), i, colorNames[index], col = colorNames[index],
srt = angle)
}
```


```
x <- seq(1, 10, 1)
xColors <- rainbow(10)
plot(x, x, xlim = c(0,11), ylim = c(0, 11), xlab = "",
ylab = "Element of rainbow(10)", type = "n",
axes = FALSE)
text(x, x, xColors, col = xColors)
axis(2, x, x)
```


```
table(Chile$region, Chile$vote)
```

```
##
##        A   N   U   Y
##    C   44 210 141 174
##    M    2  18  23  38
##    N   30 102  46 135
##    S   42 214 148 275
##    SA  69 345 230 246
```

El gráfico superior de la figura 2.21 es el resultado de aplicar la función de gráfico de barras a esta tabla de contingencia, utilizando solo las opciones predeterminadas, excepto la orientación de la etiqueta del eje parámetro, que se especifica como las = 1. Cada barra en este gráfico corresponde a un valor de la variable de voto, con la altura total de la barra representando el número de registros que enumeran cada uno de estos cuatro valores posibles.

Dentro de cada barra, la pantalla se divide en rectángulos que representan cada uno el número total de registros que enumeran el valor de voto correspondiente pero también enumeran uno de los cinco valores de región posibles.

La gráfica inferior muestra los mismos resultados que la parte superior,pero codificado por colores usando los primeros cinco de los doce colores recomendados por Iliinsky y Steele.

Además, la trama inferior también incluye una leyenda, que da la designación de región para cada color usado en la construcción de la trama. Al comparar estas dos graficos está claro que la relación entre las variables voto y región son mucho más fáciles de ver en la gráfica de colores que en la gráfica de escala de grises.

Por ejemplo, tenga en cuenta que la región "M" es la más pequeña y parece estar esencialmente ausente de la gráfica de barras "A" que representa a los votantes que pretenden abstenerse en las elecciones, y parece ser más frecuente en el diagrama de barras "Y" que representa a aquellos que tienen la intención de votar "Sí". Del mismo modo, tenga en cuenta que el diagrama codificado por colores deja en claro que "SA" (ciudad de Santiago) es generalmente la región más grande entre todos los votantes, excepto aquellos que tienen la intención de votar "Sí".

Estos detalles son difíciles, si no imposibles, de ver desde la gráfica de escala de grises.

La funcion Tableplot

La función de diagrama de tabla del paquete de diagrama de tabla proporciona caracterizaciones gráficas de conjuntos de datos con un número arbitrariamente grande de registros, brindando una vista rápida y útil de cómo se relacionan las diferentes variables. Además, esta herramienta es aplicable a conjuntos de datos mixtos con variables numéricas y categóricas, pero para representar variables categóricas, hace un uso extensivo del color.

library(tabplot) library(mlbench) data(PimaIndiansDiabetes) tableplot(PimaIndiansDiabetes) tableplot(PimaIndiansDiabetes, sortCol = “glucose”)