StatComp_11931695

Eszter Katalin Bognar

05-05-2020

#Documentation of the Data Visualization in R DataCamp course

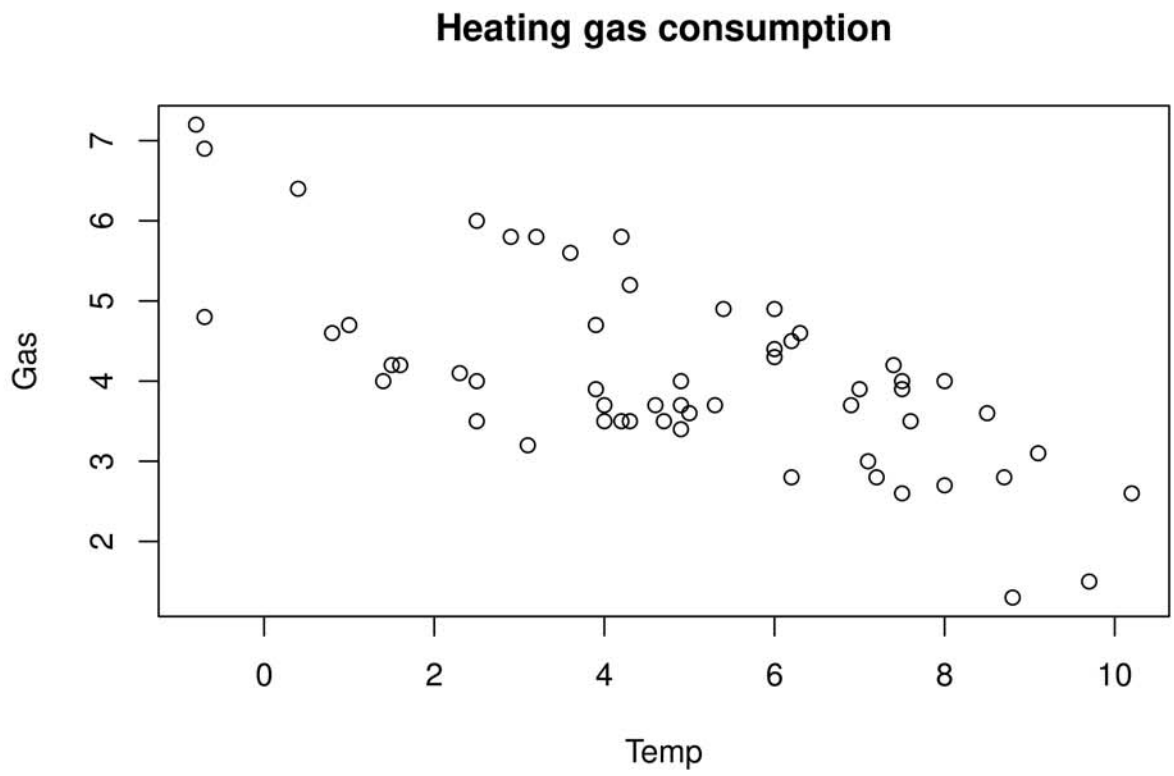
Base graphics

Simple scatterplot

```
#load library
library(MASS)
#load dataset
str(whiteside)
```

```
## 'data.frame':    56 obs. of  3 variables:
##  $ Insul: Factor w/ 2 levels "Before","After": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Temp : num  -0.8 -0.7 0.4 2.5 2.9 3.2 3.6 3.9 4.2 4.3 ...
##  $ Gas  : num  7.2 6.9 6.4 6 5.8 5.8 5.6 4.7 5.8 5.2 ...
```

```
#create scatterpplot with given x,y and title
plot(whiteside$Temp, whiteside$Gas, main="Heating gas consumption", xlab = "Temp", ylab = "Gas")
```



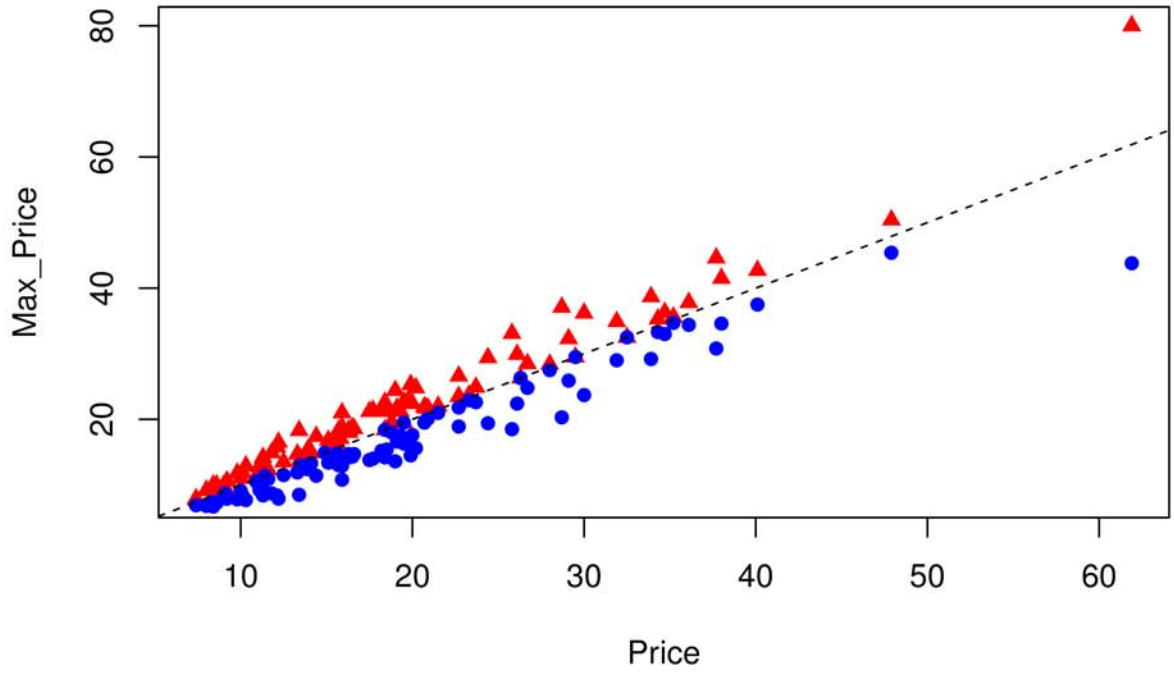
Adding details to a plot using point shapes, color, and reference lines

```
#plotting using col and pch parameters
str(Cars93)
```

```
## 'data.frame':    93 obs. of  27 variables:
##  $ Manufacturer      : Factor w/ 32 levels "Acura","Audi",..: 1 1 2 2 3 4 4 4 4 5 ...
##  $ Model             : Factor w/ 93 levels "100","190E","240",..: 49 56 9 1 6 24 54 74 73 35 ...
##  $ Type              : Factor w/ 6 levels "Compact","Large",..: 4 3 1 3 3 3 2 2 3 2 ...
##  $ Min.Price         : num  12.9 29.2 25.9 30.8 23.7 14.2 19.9 22.6 26.3 33 ...
##  $ Price             : num  15.9 33.9 29.1 37.7 30 15.7 20.8 23.7 26.3 34.7 ...
##  $ Max.Price         : num  18.8 38.7 32.3 44.6 36.2 17.3 21.7 24.9 26.3 36.3 ...
##  $ MPG.city          : int  25 18 20 19 22 22 19 16 19 16 ...
##  $ MPG.highway       : int  31 25 26 26 30 31 28 25 27 25 ...
##  $ AirBags           : Factor w/ 3 levels "Driver & Passenger",..: 3 1 2 1 2 2 2 2 2 2 ...
##  $ DriveTrain        : Factor w/ 3 levels "4WD","Front",..: 2 2 2 2 3 2 2 3 2 2 ...
##  $ Cylinders         : Factor w/ 6 levels "3","4","5","6",..: 2 4 4 4 2 2 4 4 4 5 ...
##  $ EngineSize        : num  1.8 3.2 2.8 2.8 3.5 2.2 3.8 5.7 3.8 4.9 ...
##  $ Horsepower        : int  140 200 172 172 208 110 170 180 170 200 ...
##  $ RPM               : int  6300 5500 5500 5500 5700 5200 4800 4000 4800 4100 ...
##  $ Rev.per.mile      : int  2890 2335 2280 2535 2545 2565 1570 1320 1690 1510 ...
##  $ Man.trans.avail   : Factor w/ 2 levels "No","Yes": 2 2 2 2 2 1 1 1 1 1 ...
##  $ Fuel.tank.capacity: num  13.2 18 16.9 21.1 21.1 16.4 18 23 18.8 18 ...
##  $ Passengers        : int  5 5 5 6 4 6 6 6 5 6 ...
##  $ Length            : int  177 195 180 193 186 189 200 216 198 206 ...
##  $ Wheelbase         : int  102 115 102 106 109 105 111 116 108 114 ...
##  $ Width             : int  68 71 67 70 69 69 74 78 73 73 ...
```

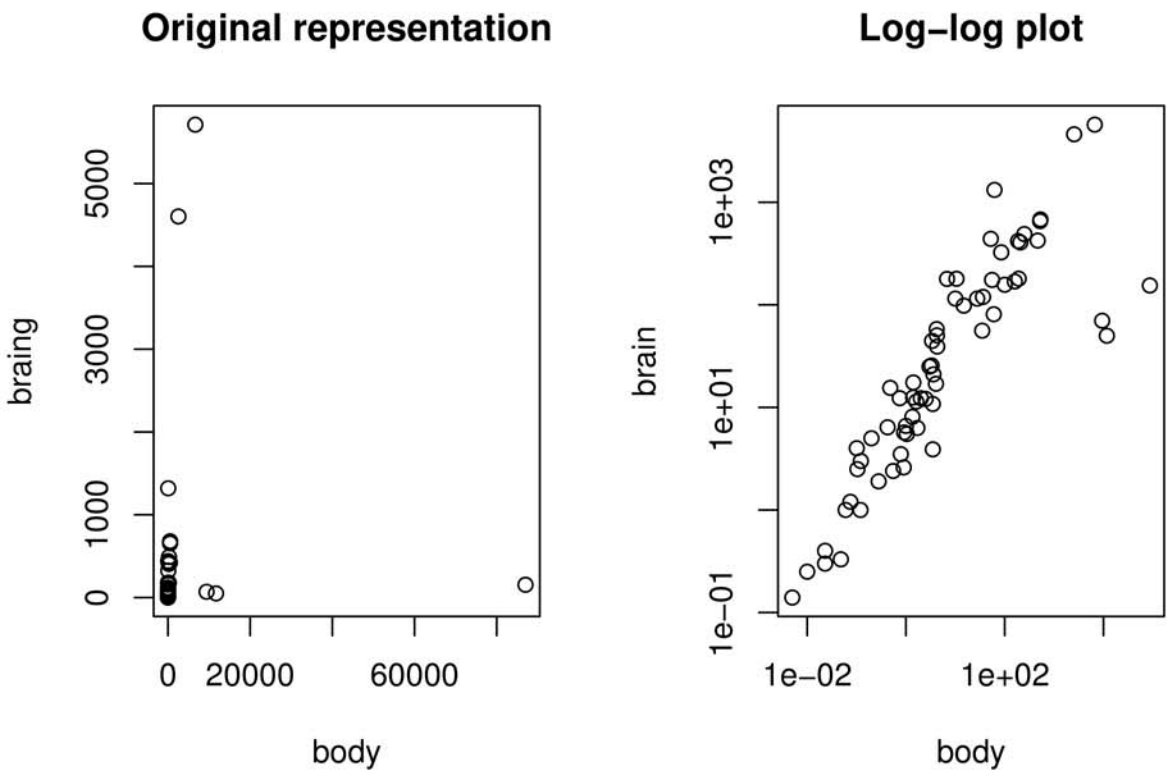
```
##  $ Turn.circle       : int  37 38 37 37 39 41 42 45 41 43 ...
##  $ Rear.seat.room    : num  26.5 30 28 31 27 28 30.5 30.5 26.5 35 ...
##  $ Luggage.room      : int  11 15 14 17 13 16 17 21 14 18 ...
##  $ Weight            : int  2705 3560 3375 3405 3640 2880 3470 4105 3495 3620 ...
##  $ Origin            : Factor w/ 2 levels "USA","non-USA": 2 2 2 2 2 1 1 1 1 1 ...
##  $ Make              : Factor w/ 93 levels "Acura Integra",..: 1 2 4 3 5 6 7 9 8 10 ...
```

```r
plot(Cars93$Price,Cars93$Max.Price,xlab = "Price", ylab = "Max_Price",pch=17,col='red')
#using the points() function to add a second set of points to your scatterplot
points(Cars93$Price,Cars93$Min.Price,xlab = "Price", ylab = "Min.Price",pch=16,col='blue')
#using abline() to to add a dashed equality reference line with intercept and slope
abline(a = 0, b = 1, lty = 2)
```



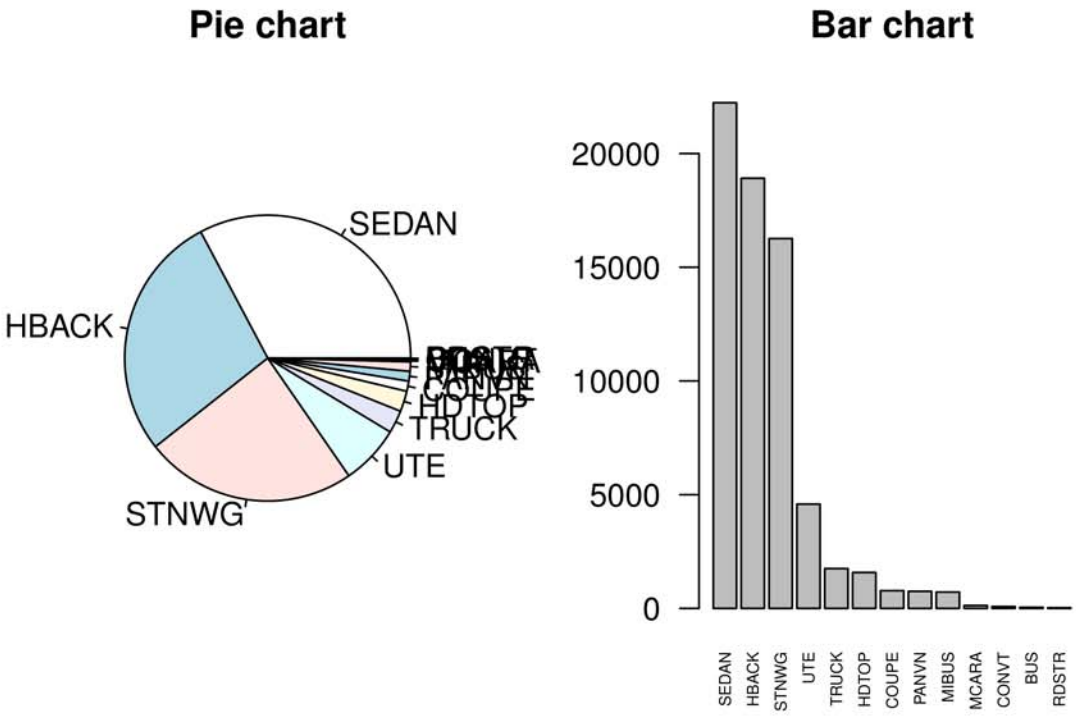
Creating multiple plot arrays

```r
library(robustbase)
#plot array with 1 row and 2 columns
par(mfrow = c(1, 2))
#adding more plots
plot(Animals2$body,Animals2$brain,xlab="body",ylab="braing")
title("Original representation")
#plotting log transform of variables
plot(Animals2$body,Animals2$brain,xlab="body",ylab="brain",log="xy")
title("Log-log plot")
```



creating table and plot it on piw and bar charts

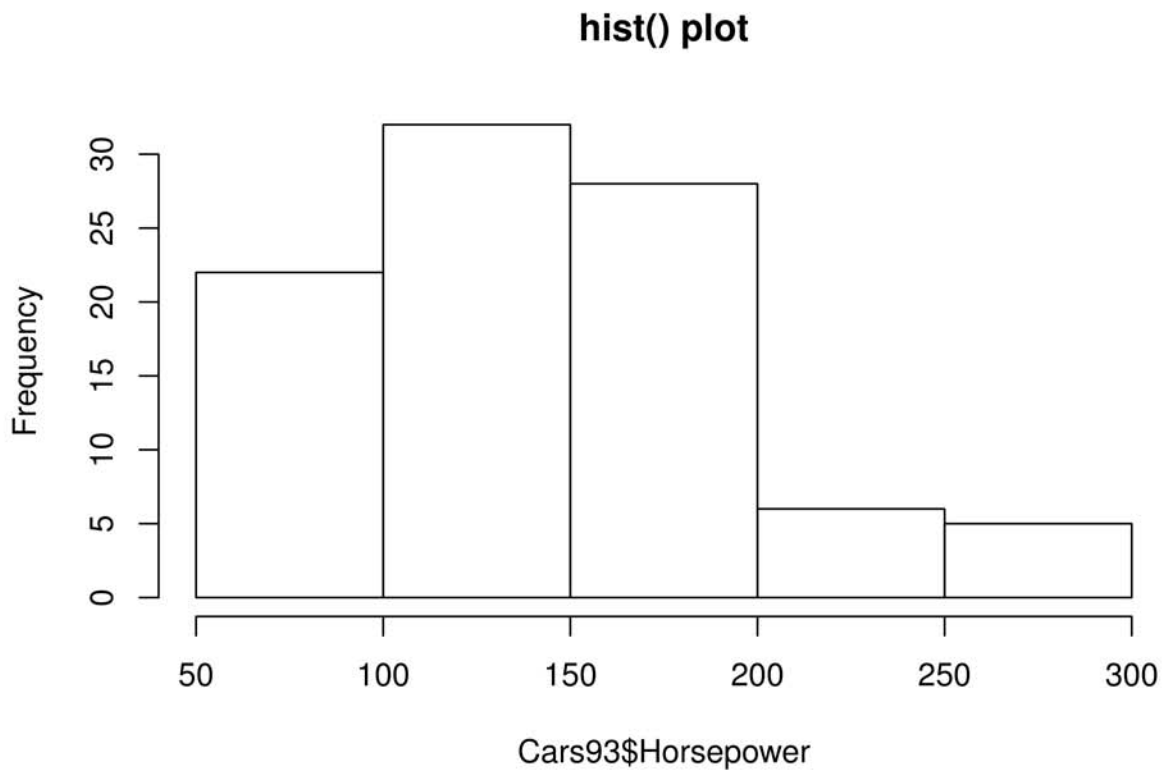
```
library(insuranceData)
data(dataCar)
par(mfrow = c(1, 2))
#create a table of record counts and sort
tbl <- sort(table(dataCar$veh_body), decreasing = TRUE)
pie(tbl)
title("Pie chart")
#barplot with perpendicular, half-sized labels
barplot(tbl, las = 2, cex.names = 0.5)
title("Bar chart")
```



Characterizing a single variable

Histogram

```
library(MASS)
hist(Cars93$Horsepower, main="hist() plot")
```

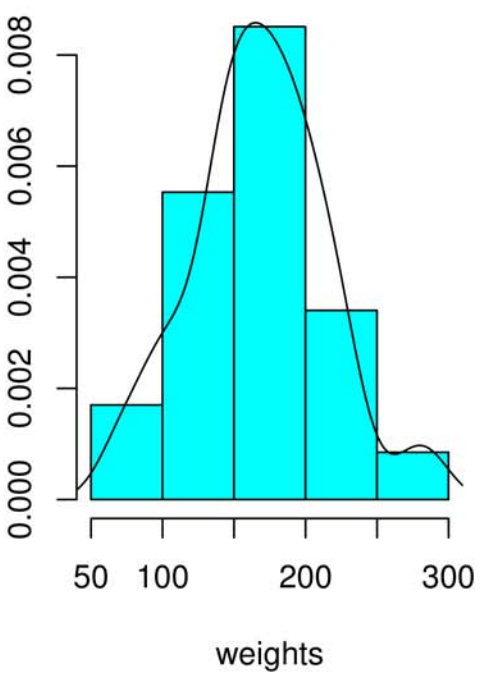


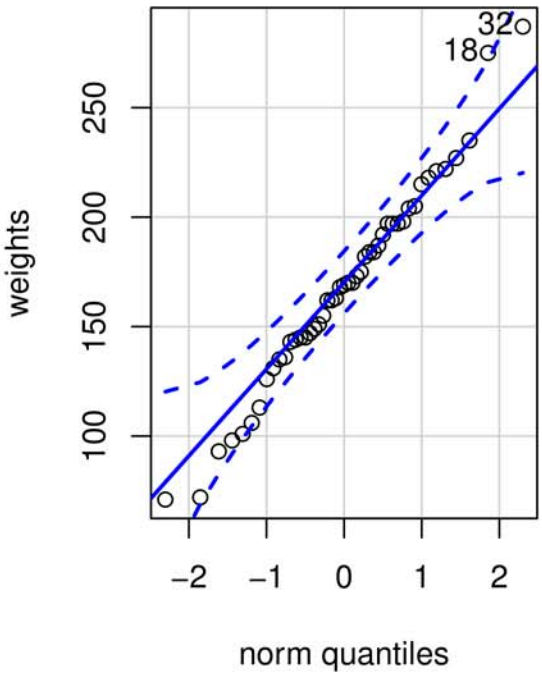
Density plot, qqPlot()

```
par(mfrow = c(1, 2))
# Create index16, pointing to 16-week chicks
index16 <- which(ChickWeight$Time == 16)
# Get the 16-week chick weights
weights <- ChickWeight$weight[index16]
# Plot the normalized histogram
truehist(weights)
# Add the density curve to the histogram
lines(density(weights))
library(car)
```

```
## Loading required package: carData
```

```
qqPlot(weights)
```



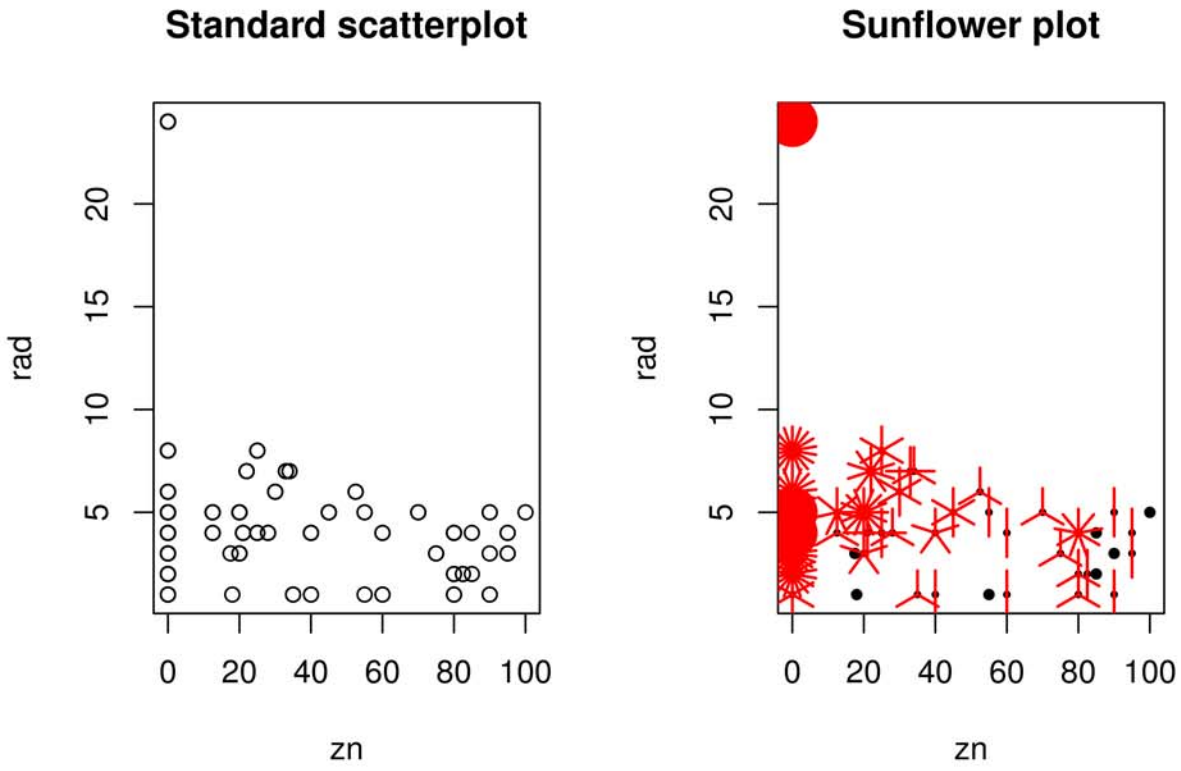



```
## [1] 32 18
```

Visualizing relations between two variables

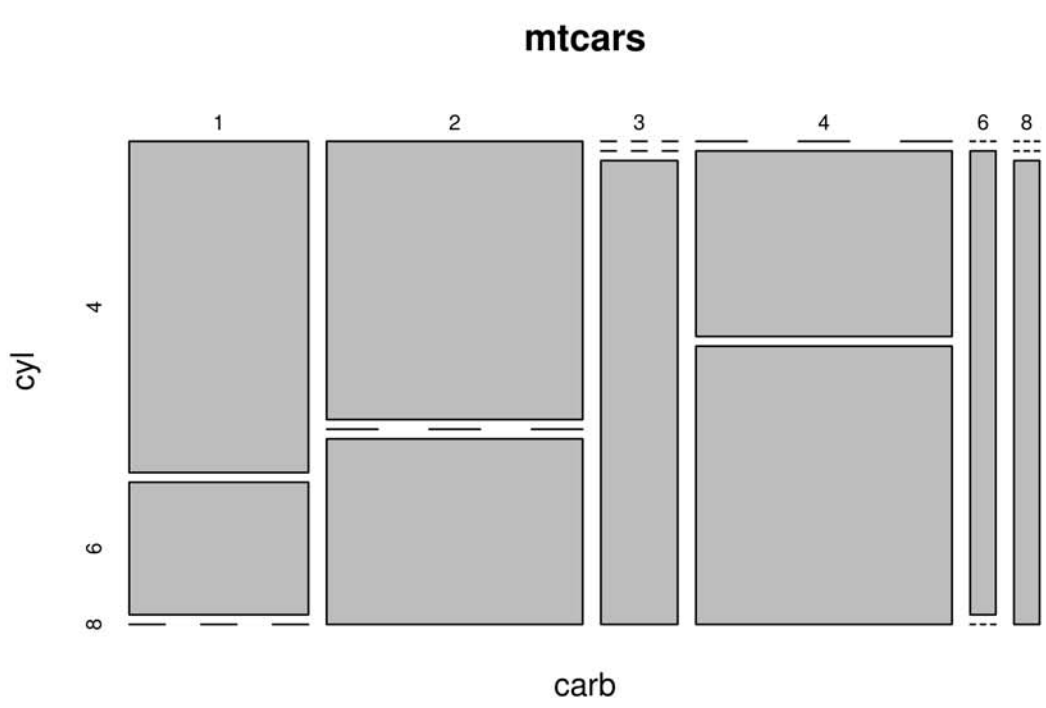
sunflowerplot() function for repeated numerical data A useful alternative that is equally effective in representing repeated data points is the sunflowerplot, which represents each repeated point by a "sunflower," with one "petal" for each repetition of a data point.

```
par(mfrow = c(1, 2))
plot(Boston$zn,Boston$rad,xlab="zn",ylab="rad")
title("Standard scatterplot")
sunflowerplot(Boston$zn,Boston$rad,xlab="zn",ylab="rad")
title("Sunflower plot")
```



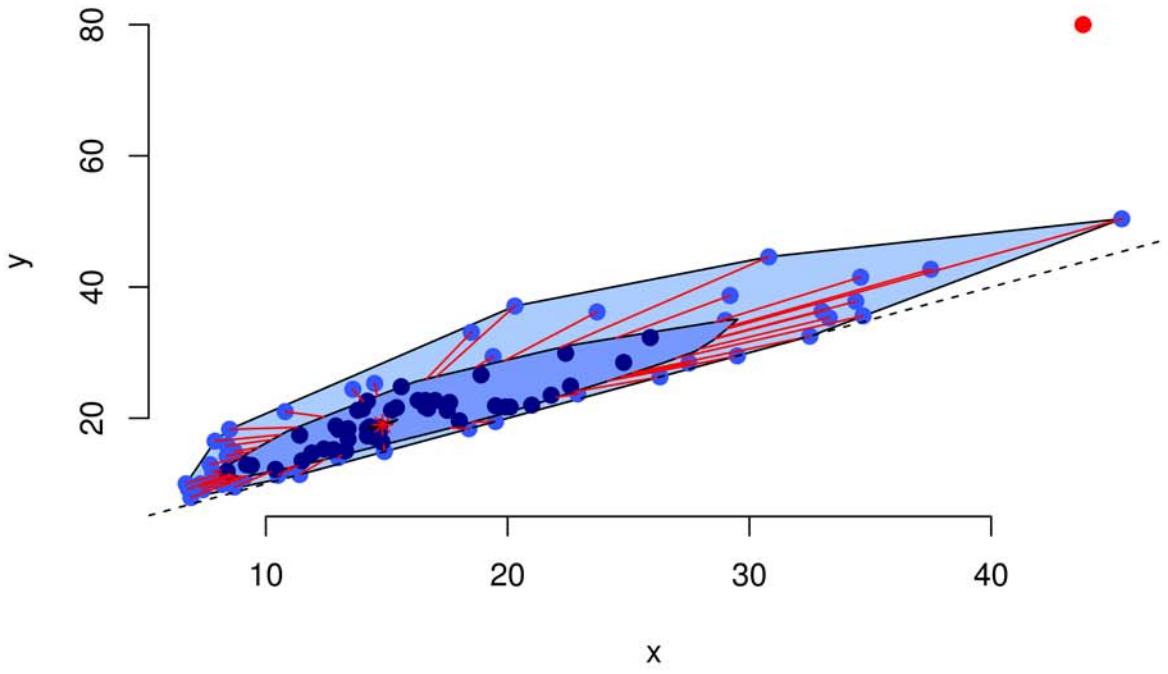
Create a mosaic plot

```
# Initialize vectors
mosaicplot(carb ~ cyl, data=mtcars)
```



bagplot

```
library(aplpack)
bagplot(Cars93$Min.Price,Cars93$Max.Price, cex=1.2)
#Add an equality reference line
abline(a = 0, b = 1, lty = 2)
```

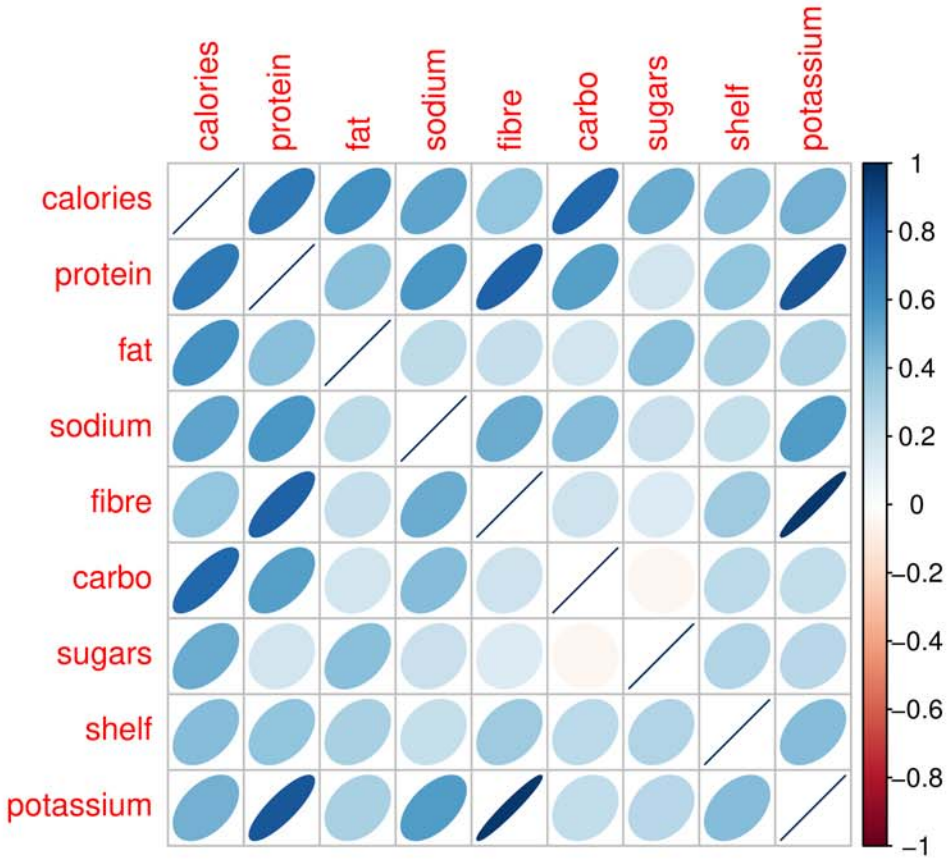


corrplot

```
library(corrplot)
```

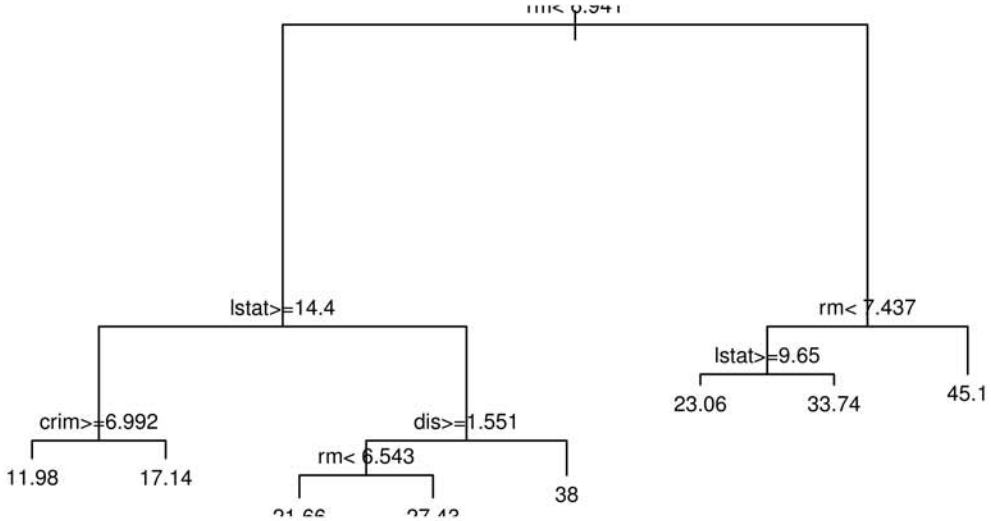
```
## corrplot 0.84 loaded
```

```
numericalVars=UScereal[2:10]
corrMat=cor(numericalVars)
corrplot(corrMat,method="ellipse")
```



Tree model

```r
# Load the rpart library
library(rpart)

# Fit an rpart model to predict medv from all other Boston variables
tree_model <- rpart(medv ~ ., data = Boston)

# Plot the structure of this decision tree model
plot(tree_model)

# Add labels to this plot
text(tree_model, cex = 0.7)
```



The plot() function and its options

using the par function

```
# Assign the return value from the par() function to plot_pars
plot_pars=par()

# Display the names of the par() function's list elements
names(plot_pars)
```

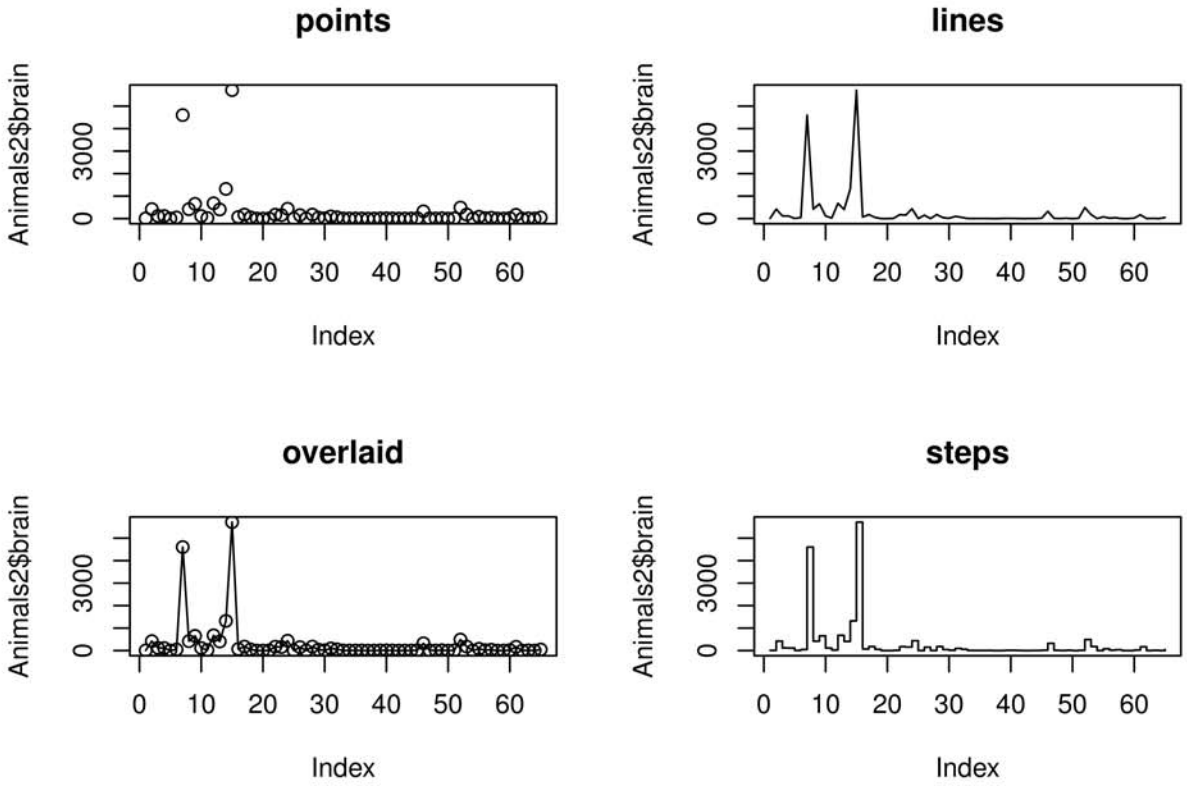
```
##  [1] "xlog"      "ylog"      "adj"       "ann"       "ask"       "bg"
##  [7] "bty"       "cex"       "cex.axis"  "cex.lab"   "cex.main"  "cex.sub"
## [13] "cin"       "col"       "col.axis"  "col.lab"   "col.main"  "col.sub"
## [19] "cra"       "crt"       "csi"       "cxy"       "din"       "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"
```

```r
# Display the number of par() function list elements
length(names(plot_pars))
```

```
## [1] 72
```

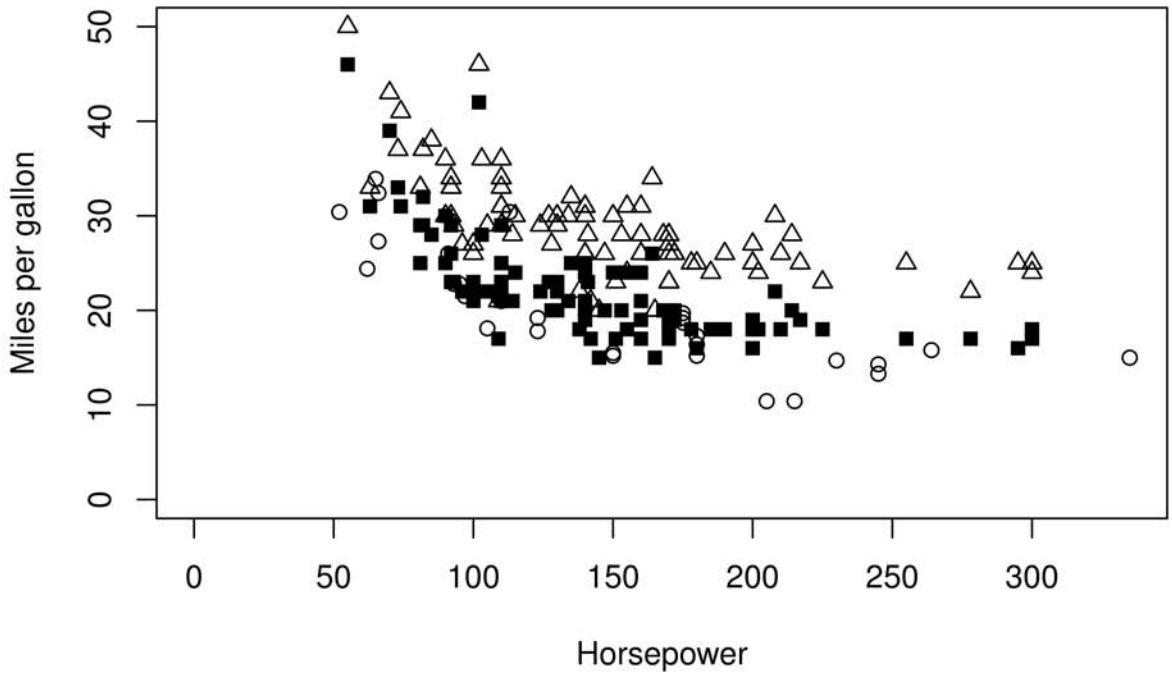
using the type function

```r
library(robustbase)
# Set up a 2-by-2 plot array
par(mfrow = c(2, 2))

# Plot the Animals2 brain weight data as points
plot(Animals2$brain, type = "p")

# Add the title
title("points")

# Plot the brain weights with lines
plot(Animals2$brain, type = "l")

# Add the title
title("lines")

# Plot the brain weights as lines overlaid with points
plot(Animals2$brain, type = "o")

# Add the title
title("overlaid")

# Plot the brain weights as steps
plot(Animals2$brain, type = "s")

# Add the title
title("steps")
```



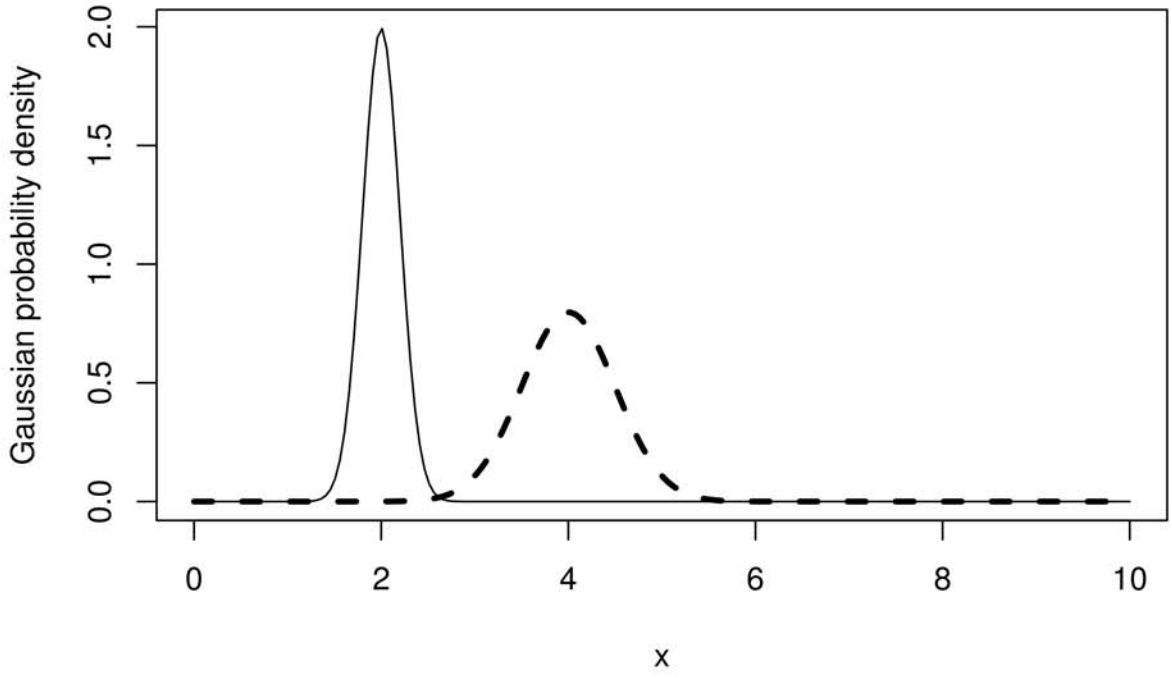
plotting data from multiple sources on a common set of axes

```r
# Compute max_hp
max_hp <- max(Cars93$Horsepower, mtcars$hp)

# Compute max_mpg
max_mpg <- max(Cars93$MPG.city, Cars93$MPG.highway,
               mtcars$mpg)

# Create plot with type = "n"
plot(Cars93$Horsepower, Cars93$MPG.city,
     type = "n", xlim = c(0, max_hp),
     ylim = c(0, max_mpg), xlab = "Horsepower",
     ylab = "Miles per gallon")

# Add open circles to plot
points(mtcars$hp, mtcars$mpg, pch = 1)

# Add solid squares to plot
points(Cars93$Horsepower, Cars93$MPG.city,
       pch = 15)

# Add open triangles to plot
points(Cars93$Horsepower, Cars93$MPG.highway,
       pch = 2)
```



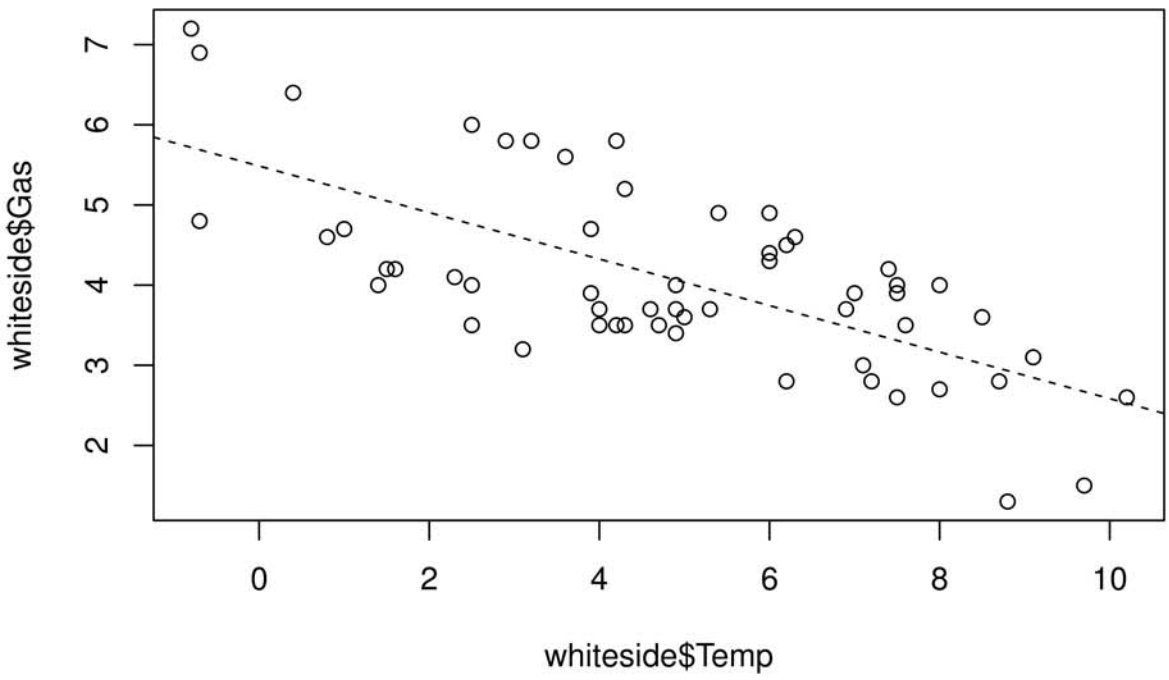
Adding lines and points to plots The lines() function and line types

```r
# Create the numerical vector x
x <- seq(0, 10, length = 200)

# Compute the Gaussian density for x with mean 2 and standard deviation 0.2
gauss1 <- dnorm(x, mean = 2, sd = 0.2)

# Compute the Gaussian density with mean 4 and standard deviation 0.5
gauss2 <- dnorm(x, mean = 4, sd = 0.5)

# Plot the first Gaussian density
plot(x,gauss1,type="l",ylab="Gaussian probability density")

# Add lines for the second Gaussian density
lines(x, gauss2,lty=2, lwd=3)
```



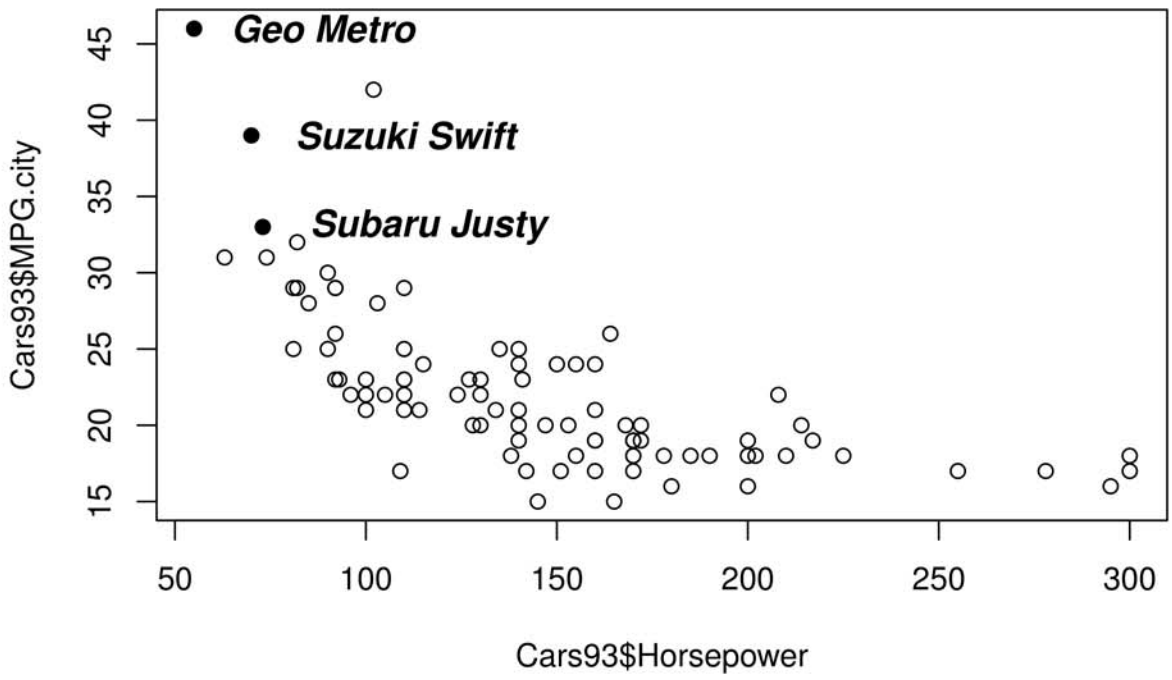
plotting linar trendline using linear regression

```
# Build a linear regression model for the whiteside data
linear_model=lm(Gas ~ Temp,data=whiteside)

# Create a Gas vs. Temp scatterplot from the whiteside data
plot(whiteside$Temp, whiteside$Gas)

# Use abline() to add the linear regression line
abline(linear_model, lty = 2)
```



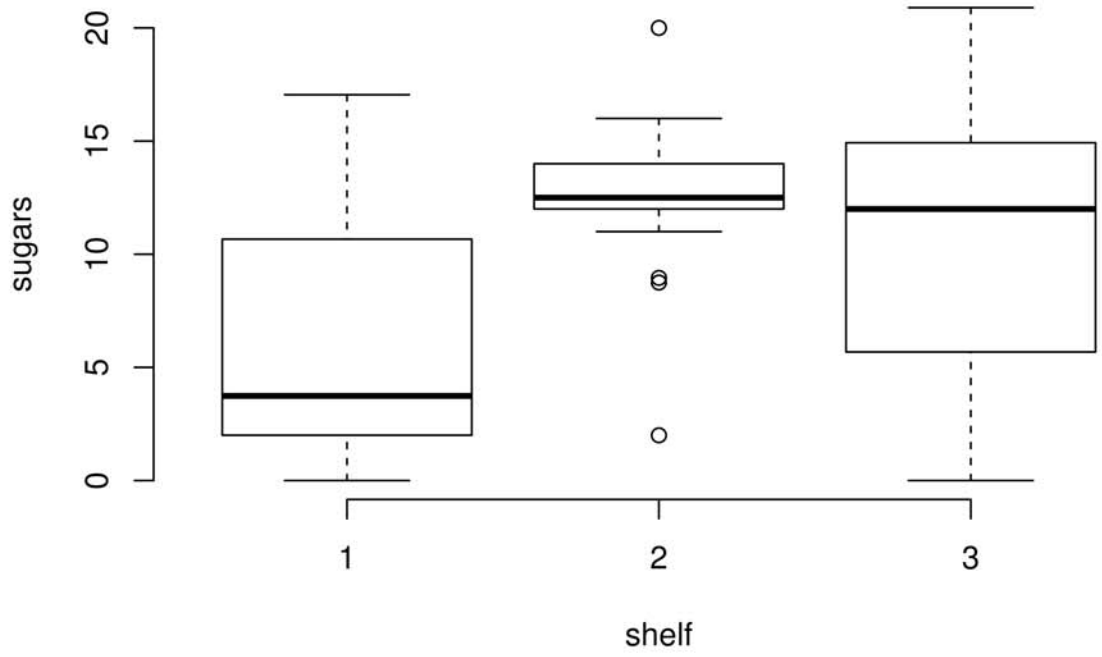
Adjusting text position, size, and font

```
# Plot MPG.city vs. Horsepower as open circles
plot(Cars93$Horsepower, Cars93$MPG.city)

# Create index3, pointing to 3-cylinder cars
index3 <- which(Cars93$Cylinders == 3)

# Highlight 3-cylinder cars as solid circles
points(Cars93$Horsepower[index3],
       Cars93$MPG.city[index3],
       pch = 16)

# Add car names, offset from points, with larger bold text
text(x = Cars93$Horsepower[index3],
     y = Cars93$MPG.city[index3],
     labels = Cars93$Make[index3], adj = -0.2, cex=1.2,font = 4)
```



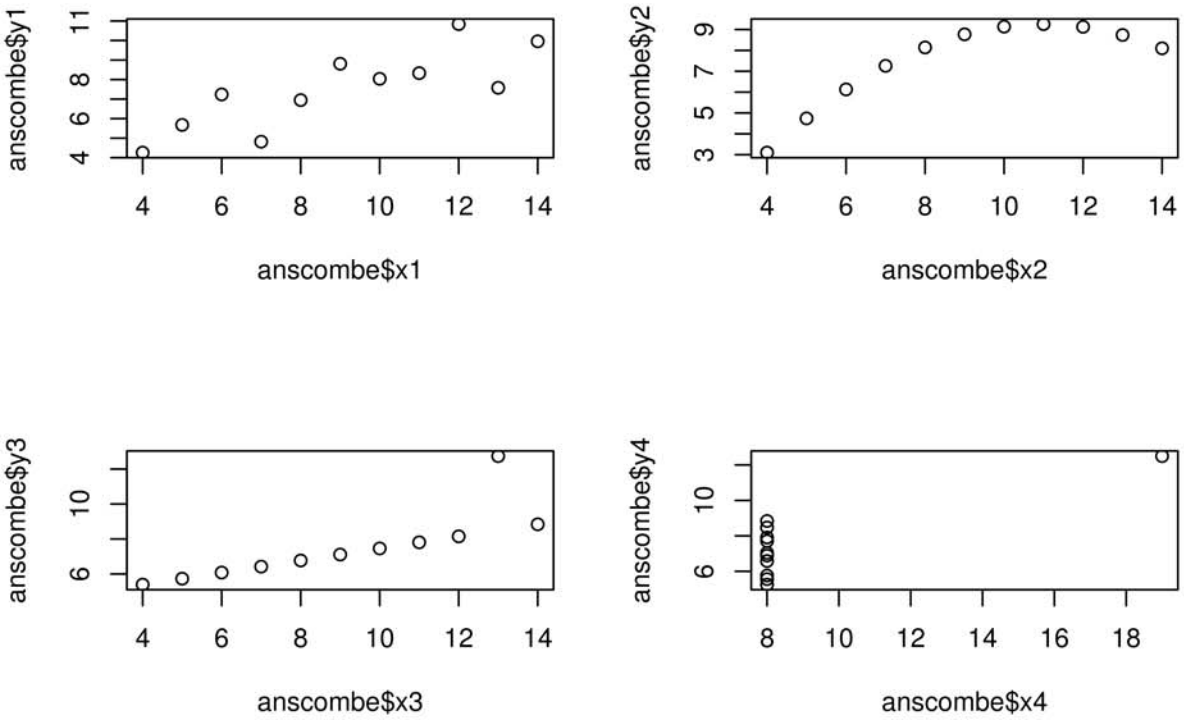
Custom axis

```
# Create a boxplot of sugars by shelf value, without axes
boxplot(sugars ~ shelf, data = UScereal,
        axes = FALSE)

# Add a default y-axis to the left of the boxplot
axis(side = 2)

# Add an x-axis below the plot, labelled 1, 2, and 3
axis(side = 1, at = c(1, 2, 3))
```



Managing visual complexity Creating plot arrays

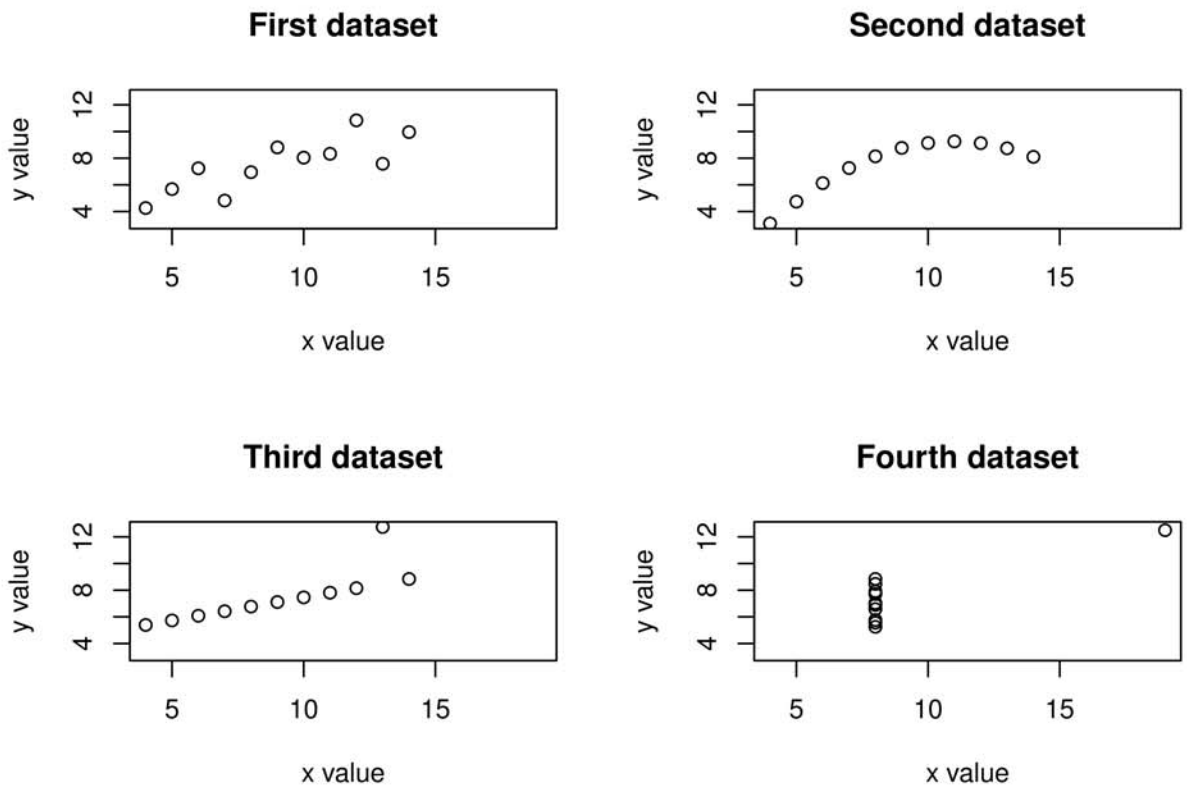
```
# Set up a two-by-two plot array
par(mfrow = c(2, 2))

# Plot y1 vs. x1
plot(anscombe$x1, anscombe$y1)

# Plot y2 vs. x2
plot(anscombe$x2, anscombe$y2)

# Plot y3 vs. x3
plot(anscombe$x3, anscombe$y3)

# Plot y4 vs. x4
plot(anscombe$x4, anscombe$y4)
```



common scaling and individual titles

```
# Define common x and y limits for the four plots
xmin <- min(anscombe$x1, anscombe$x2, anscombe$x3, anscombe$x4)
xmax <- max(anscombe$x1, anscombe$x2, anscombe$x3, anscombe$x4)
ymin <- min(anscombe$y1, anscombe$y2, anscombe$y3, anscombe$y4)
ymax <- max(anscombe$y1, anscombe$y2, anscombe$y3, anscombe$y4)

# Set up a two-by-two plot array
par(mfrow = c(2, 2))

# Plot y1 vs. x1 with common x and y limits, labels & title
plot(anscombe$x1, anscombe$y1,
     xlim = c(xmin, xmax),
     ylim = c(ymin, ymax),
     xlab = "x value", ylab = "y value",
     main = "First dataset")

# Do the same for the y2 vs. x2 plot
plot(anscombe$x2, anscombe$y2,
     xlim = c(xmin, xmax),
     ylim = c(ymin, ymax),
     xlab = "x value", ylab = "y value",
     main = "Second dataset")

# Do the same for the y3 vs. x3 plot
plot(anscombe$x3, anscombe$y3,
```

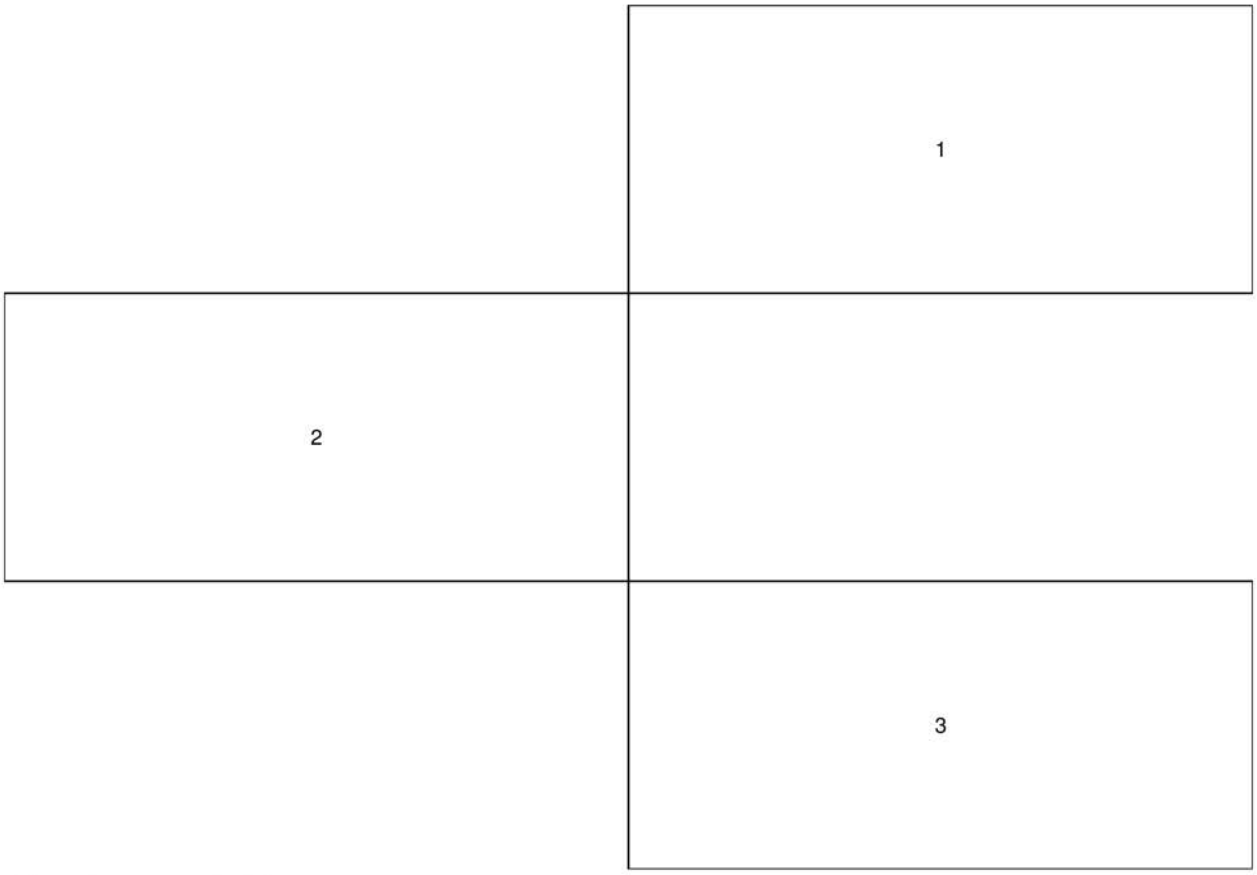
```
     xlim = c(xmin, xmax),
     ylim = c(ymin, ymax),
     xlab = "x value", ylab = "y value",
     main = "Third dataset")

# Do the same for the y4 vs. x4 plot
plot(anscombe$x4, anscombe$y4,
     xlim = c(xmin, xmax),
     ylim = c(ymin, ymax),
     xlab = "x value", ylab = "y value",
     main = "Fourth dataset")
```



layout matrices

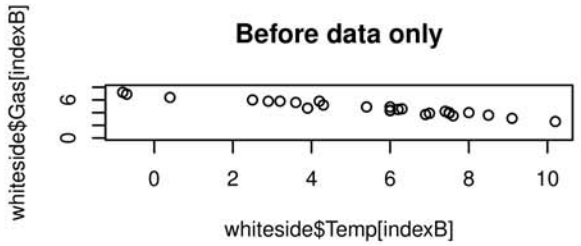
```
# Use the matrix function to create a matrix with three rows and two columns
layoutMatrix <- matrix(
  c(
    0, 1,
    2, 0,
    0, 3
  ),
  byrow = TRUE,
  nrow = 3
)
# Call the layout() function to set up the plot array
layout(layoutMatrix)
```

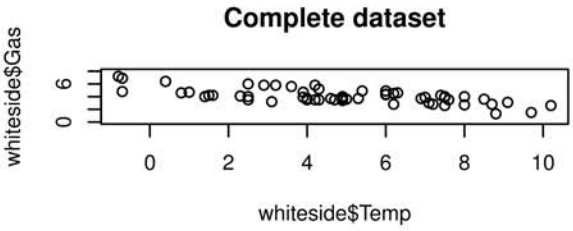
```
# Show where the three plots will go
layout.show(3)
```

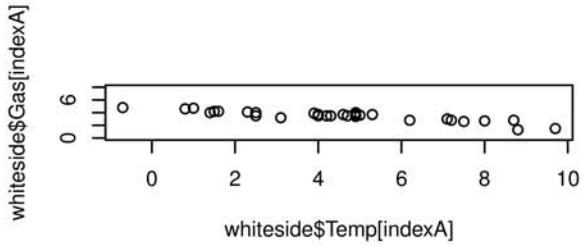


triangular array of plots

```
# Set up the plot array
layout(layoutMatrix)

# Construct the vectors indexB and indexA
indexB <- which(whiteside$Insul == "Before")
indexA <- which(whiteside$Insul == "After")

# Create plot 1 and add title
plot(whiteside$Temp[indexB], whiteside$Gas[indexB],
     ylim = c(0, 8))
title("Before data only")

# Create plot 2 and add title
plot(whiteside$Temp, whiteside$Gas,
     ylim = c(0, 8))
title("Complete dataset")

# Create plot 3 and add title
plot(whiteside$Temp[indexA], whiteside$Gas[indexA],
     ylim = c(0, 8))
```

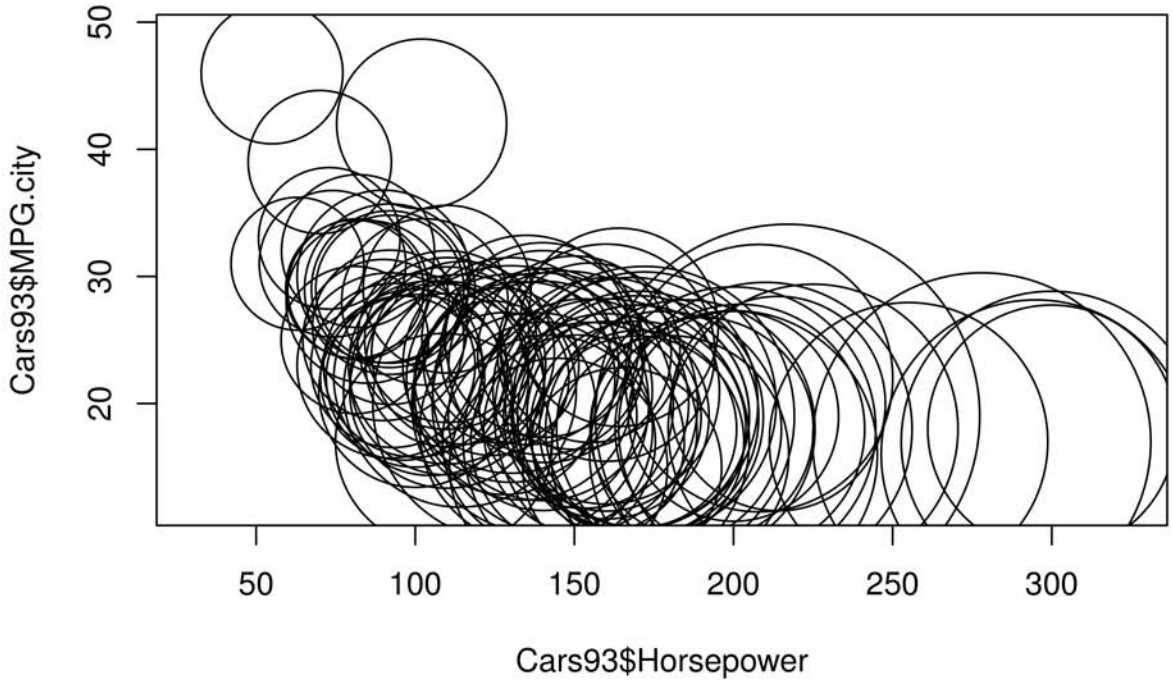






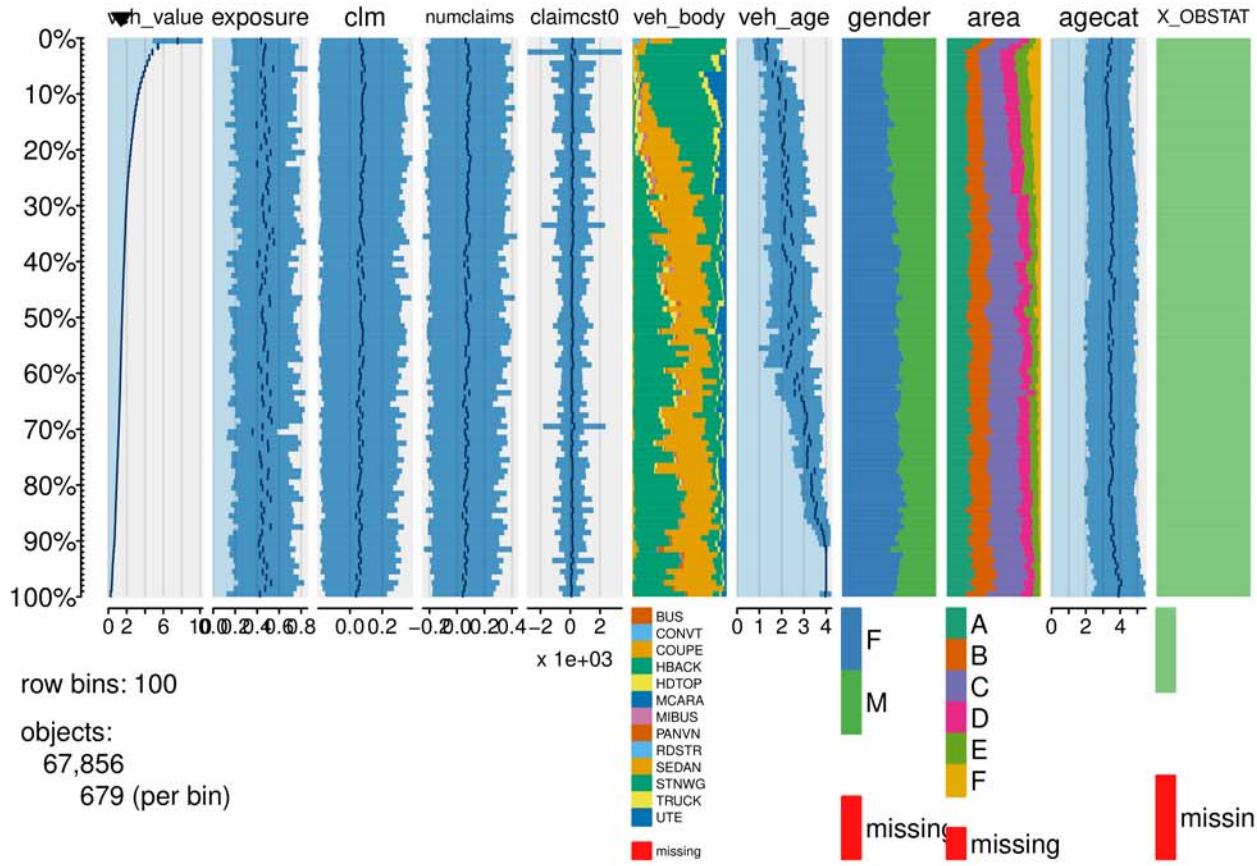
##Creating and saving complex plots Using the symbols() function to display relations between more than two variables

```
# Call symbols() to create the default bubbleplot
symbols(Cars93$Horsepower, Cars93$MPG.city,
        circles = sqrt(Cars93$Price))

# Repeat, with the inches argument specified
symbols(Cars93$Horsepower, Cars93$MPG.city, circles = sqrt(Cars93$Price))
```



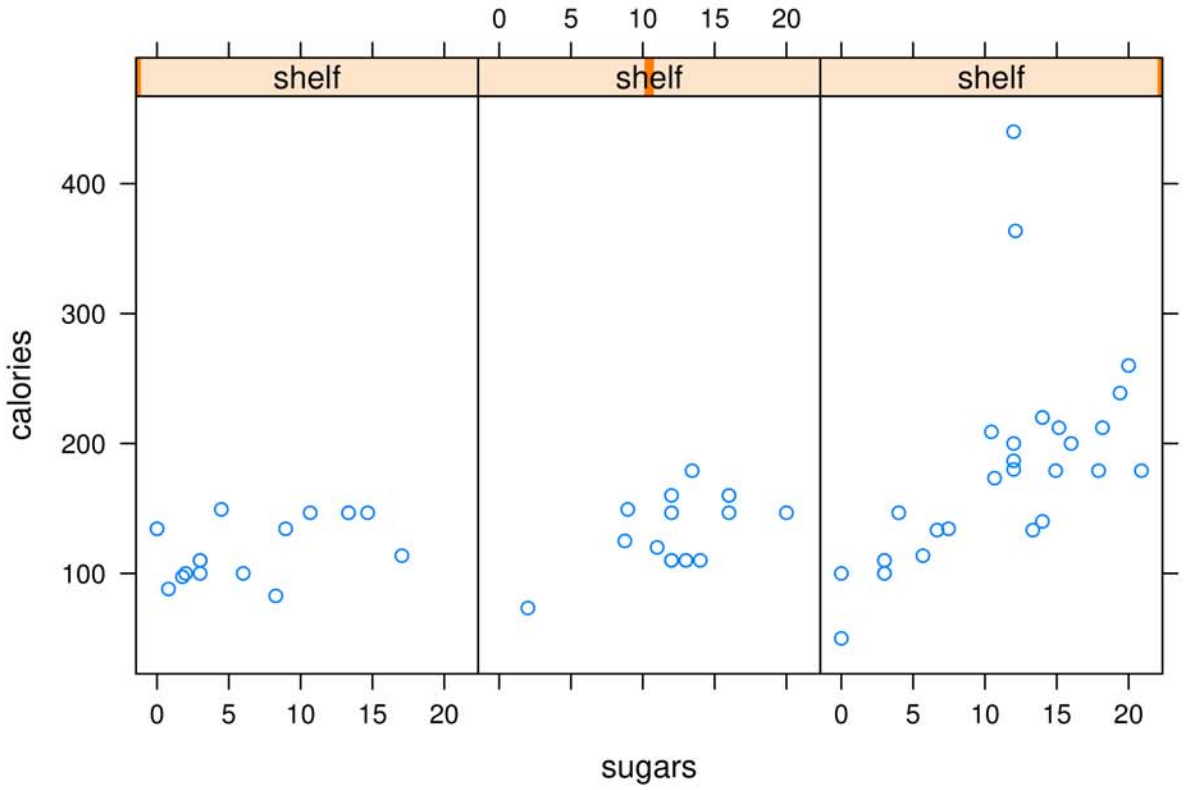
###Other useful packages Tabplot

```
# Load the insuranceData package
library(insuranceData)

# Use the data() function to load the dataCar data frame
data(dataCar)

# Load the tabplot package
suppressPackageStartupMessages(library(tabplot))


# Generate the default tableplot() display
tableplot(dataCar)
```



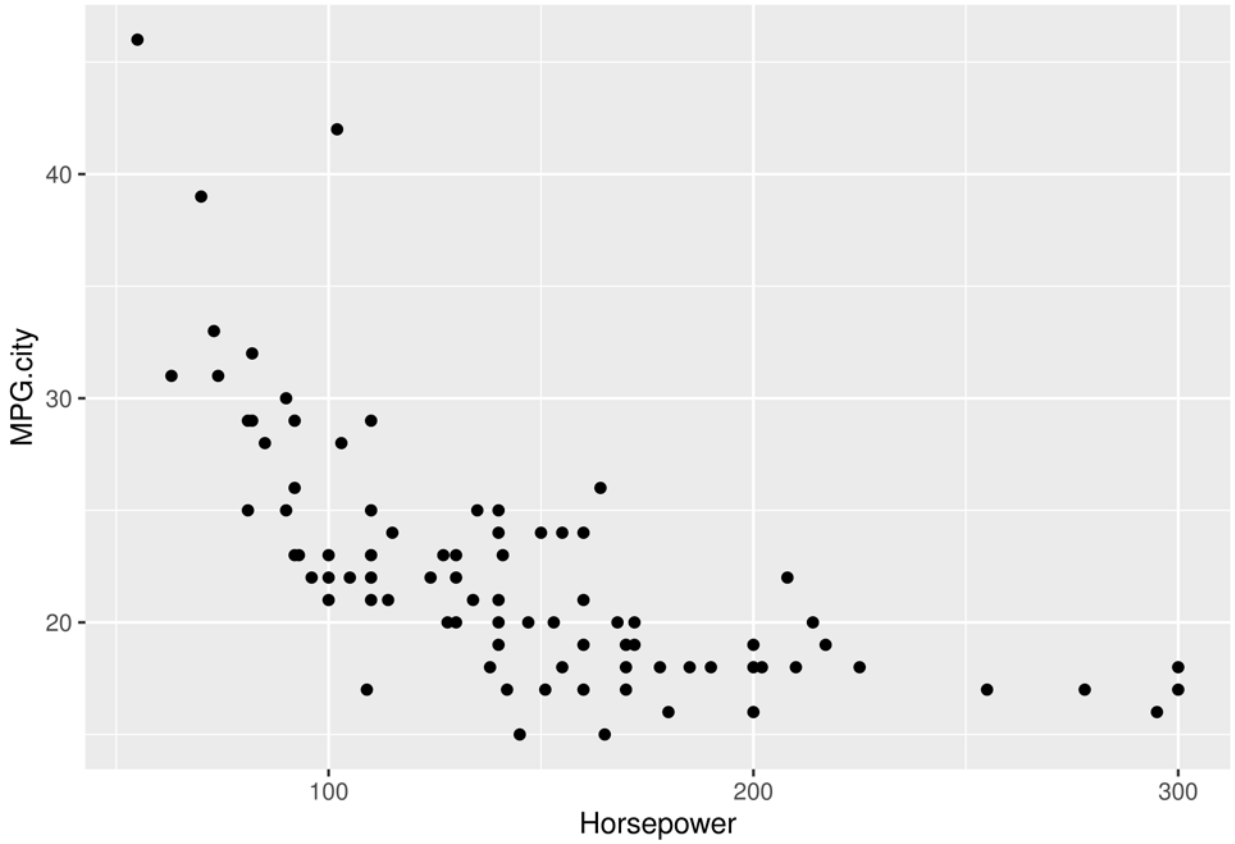
Lattice

```
# Load the lattice package
library(lattice)
# Construct the formula
calories_vs_sugars_by_shelf <- calories ~ sugars | shelf

# Use xyplot() to draw the conditional scatterplot
xyplot(calories_vs_sugars_by_shelf,UScereal)
```



ggplot2

```
# Load the ggplot2 package
library(ggplot2)

# Create the basic plot (not displayed): basePlot
basePlot <- ggplot(Cars93, aes(x = Horsepower, y = MPG.city))

# Display the basic scatterplot
basePlot +
  geom_point()
```


MPG.city
40
30
20
100
200
300
Horsepower