Chapter 3 Graphical Displays Part 2

Tanweer Shapla

2023-10-10

## Convex hull

A hull means a covering or an envelop. The convex hull is a lot like wrapping an irregularly shaped object with the smallest amount of gift paper: The most extreme points in any direction will define the ultimate shape of the object/package.

Mathematically, the convex hull is the smallest convex polygon containing all of the data points.

Convex hulls have many real-life applications, such as in computer graphics, image processing, pattern recognition, collision detection, robotics, astronomy, etc. Here are a few examples:

In computer graphics, convex hulls are used to create 3D models of objects by wrapping a surface around a set of points. This is often used in computer-aided design (CAD) software to create models of buildings, vehicles, and other objects.

In image processing, convex hulls are used to segment images by identifying the shape of objects in the image. This is often used in medical imaging to identify tumors or other abnormalities, or in industrial inspection to identify defects in products.

In pattern recognition, convex hulls are used to identify patterns in data. This is often used in finance to identify patterns in stock prices, or in manufacturing to identify patterns in production data.

In collision detection, convex hulls are used to identify when two objects are colliding. This is often used in video games and simulations to determine when characters or vehicles collide.

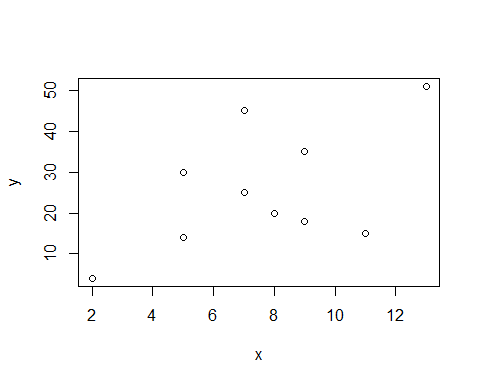
In robotics, convex hulls are used to identify the shape of objects and plan paths around them. This is often used in autonomous vehicles and drones to navigate through obstacles.

In astronomy, astronomers use convex hulls to analyze the shapes of galaxies and star clusters.

## Drawing a convex hull for two-dimensional data

In R, the chull() function returns the indices of the data points that are most extreme in each direction:

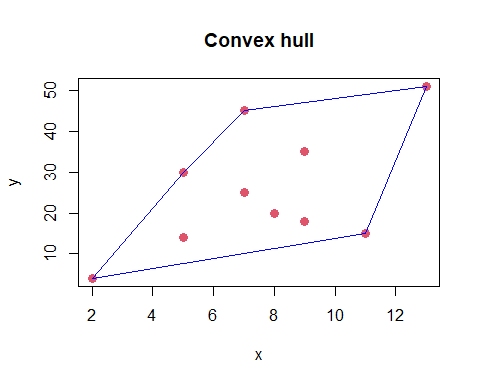
x=c(2, 5, 8, 9, 11, 13, 5, 7, 9, 7)  
y=c(4, 14, 20, 18, 15, 51, 30, 45, 35, 25)  
plot(x,y)



extr=chull(cbind(x,y))#chull() gives the indices of extreme points in each direction  
cbind(x[extr], y[extr]) #gives extreme points

[,1] [,2]  
[1,] 11 15  
[2,] 2 4  
[3,] 5 30  
[4,] 7 45  
[5,] 13 51

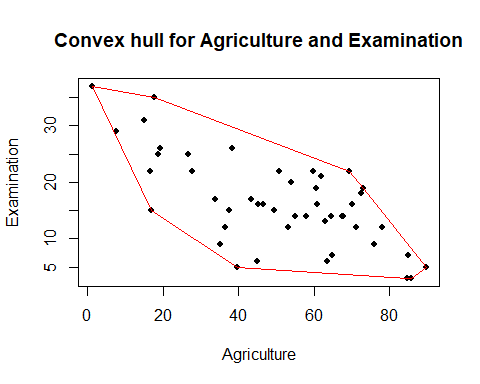
extr1 <- c(extr, extr[1])  
plot(x,y, pch = 16, col = 2, cex = 1.25, main="Convex hull")  
lines(x[extr1], y[extr1], type = "l", col = "blue", lwd = 1)



The lines command adds lines to the figure with the type=“l” option. The lines that are drawn also need to return back to the beginning, so the list of indices extr from the chull() must repeat the first observation at the end of this list in order to complete the polygon. The object extr1 does it.

## Drawing convex hull for swiss data

attach(swiss)  
#extr.swiss=chull(cbind(swiss$Agriculture, swiss$Examination))  
extr.s=chull(cbind(Agriculture, Examination))  
extr.s=c(extr.s, extr.s[1])  
plot(Agriculture, Examination, pch = 16, col = 1, cex = 0.8, main="Convex hull for Agriculture and Examination")  
lines(Agriculture[extr.s], Examination[extr.s], type="l", col="red", lwd=1.2)



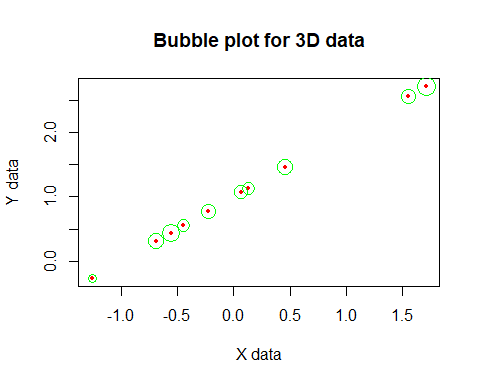
detach(swiss)

## Drawing a Bubble plot for 3 dimensional data

In a bubble plot, a two dimensional scatterplot is drawn and then the third variable is shown by drawing a circle around each point. The size of the circle or bubble represents value of the third variable.

set.seed(123) #initialize a random number generator  
x=rnorm(10)#generating 10 random numbers from a standard normal dist  
y<-x+1  
z=rnorm(10,1,0.2)  
d=data.frame(x,y,z)

plot(x,y, xlab="X data", ylab="Y data",   
 main="Bubble plot for 3D data", pch=16, col="red", cex=0.5)  
symbols(d$x, d$y, circles=d$z, inches = 0.1, add = TRUE, lwd = 0.5, fg = "green")



We can use with() function in the last line above: with(d,symbols(x, y, circles=z, inches = .1, add = TRUE, lwd = 0.5, fg = “green”))

Note that the size of the circle is proportional to the magnitude of the third variable,

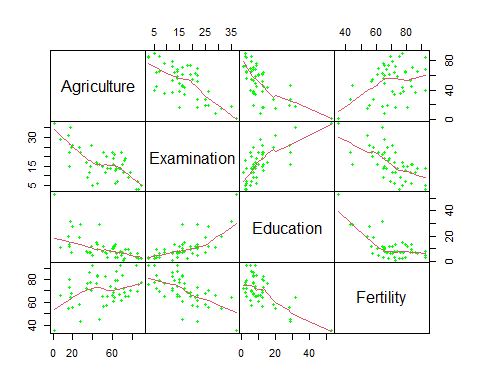
inches= sets the scale of the size of the circle,

add=T shows the (x,y) points on graph,

lwd determines the thickness of lines to draw circles.

## Drawing matrix of scatterplots for multidimensional data

pairs(cbind(Agriculture=swiss$Agriculture, Examination=swiss$Examination, Education=swiss$Education, Fertility=swiss$Fertility), panel=panel.smooth, col="green",gap = 0, lwd = 1, pch = 16, cex = 0.6)



Variable in a box goes vertically for that row, the names of the variables runs down the diagonal. Every possible pair of these variables is plotted twice with the axes reversed above and below the diagonal.

Panel option draws loess regression. A loess curve, short for “locally weighted scatterplot smoothing,” is a statistical technique used in data analysis and regression modeling to create a smooth curve that captures the underlying trend in a scatterplot of data points. It is particularly useful when you have noisy data or when the relationship between variables is not linear. Instead of fitting a single global regression line to all the data points, a loess curve performs local regression. This means that for each point on the X-axis, a subset of nearby data points is selected based on a smoothing parameter (bandwidth or span). The selected points are weighted based on their proximity to the point of interest, with closer points receiving higher weights. Once the local data points and their weights are determined, a weighted regression model, often a low-degree polynomial is fitted to the selected points. This regression model captures the local trend in the data. The loess curve is created by repeating this local regression process for all points along the X-axis, producing a smooth curve that best represents the underlying trend in the data.

Loess curves are non-parametric approaches commonly used in exploratory data analysis to visualize trends and patterns in data before more formal statistical modeling. They are also useful for making predictions or estimating values between data points when a smooth trend is required.

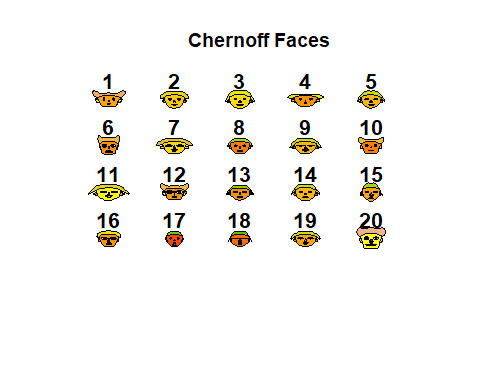
In pairs() function, the argument gap=0 removes the spacing between the panels,

xaxt = “n”, yaxt = “n” removes the numbers in the axes.

## Drawing Chernoff faces

Chernoff (1973) used cartoon faces to demonstrate multivariate information. Each variable can represent some aspect of the face: head size and color, degree of smile or frown, amount and shape of hair, the spacing and size of eyes, pupil position, and so on.

#install.packages("aplpack") #aplpack stands for another plot package  
library(aplpack)   
set.seed(1234)  
age=rnorm(20, 50, 2) # 20 random values from N(50,2)  
weight=rnorm(20, 110, 3)  
height=70\*runif(20, 1,2)# 20 values from U(1,2) multiplied by 70  
bp= rnorm(20, 122, 2)  
data=round(data.frame(age, weight, height, bp), digits=2)  
faces(data, main="Chernoff Faces")



## effect of variables:  
## modified item Var   
## "height of face " "age"   
## "width of face " "weight"  
## "structure of face" "height"  
## "height of mouth " "bp"   
## "width of mouth " "age"   
## "smiling " "weight"  
## "height of eyes " "height"  
## "width of eyes " "bp"   
## "height of hair " "age"   
## "width of hair " "weight"  
## "style of hair " "height"  
## "height of nose " "bp"   
## "width of nose " "age"   
## "width of ear " "weight"  
## "height of ear " "height"

#As many as 18 variables can be described using different face features   
  
in.max.wt=match(max(weight), weight)#gives the index of first value in the second  
in.max.wt

## [1] 11