STAT 210
Applied Statistics and Data Analysis:
Graphics in R
High-level commands

Joaquín Ortega

joaquin.ortegasanchez@kaust.edu.sa

`curve`

curve

This function graphs a function in a given interval:

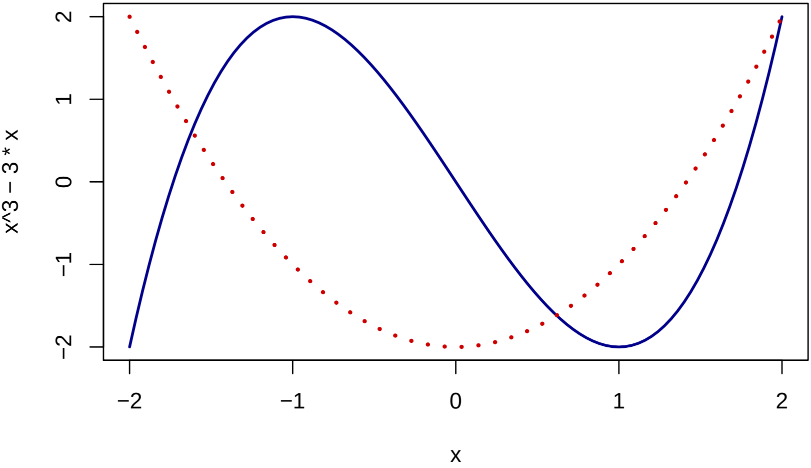
```
curve(expr, from, to, add = FALSE, ...)
```

where

- `expr` function with variable `x`
- `from, to` Interval limits
- `add` Logical. If `TRUE` the graph is added to the active window.

curve

```r
curve(x^3 - 3*x, -2,2,lwd=2, col='darkblue')
curve(x^2 -2 , add= TRUE, col='red3', lwd=3, lty = 3)
```



Boxplots

Boxplots

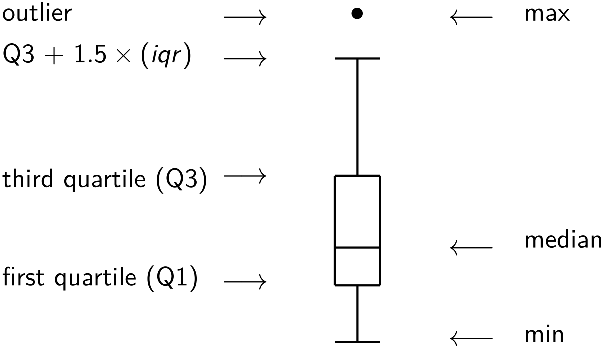
Boxplots were proposed by John W. Tukey in the 1970s as a quick way to visualize the main features of a data set.

There are several versions, but in general the interquartile range (iqr) is represented by a box or rectangle, so that the ends of the rectangle are located in the first and third quartiles, as shown in Figure 4.1.

Inside the rectangle, the location of the median is indicated by a line or point. Outside the rectangle, two segments are drawn, called 'whiskers', which reach the furthest data at a distance less than or equal to $1.5 \times$ (iqr) from the rectangle.

Any point that is not included in this range is represented individually and is a potential outlier.

Boxplots



outlier

Q3 $+ 1.5 \times (iqr)$

third quartile (Q3)

first quartile (Q1)

max

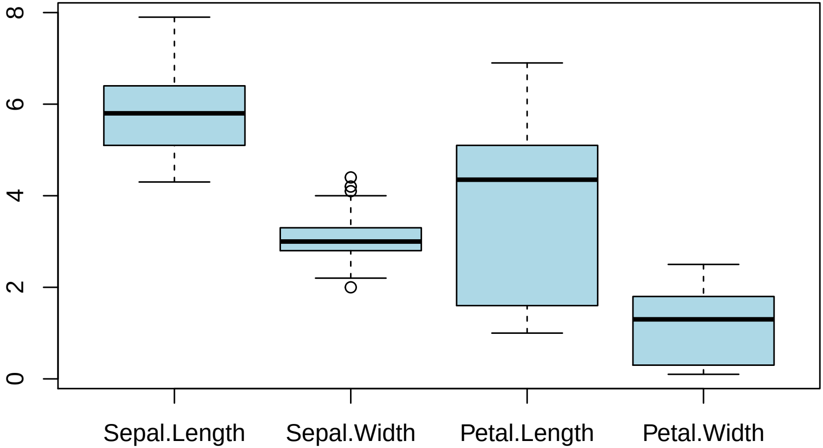
median

min

Figure 4.1

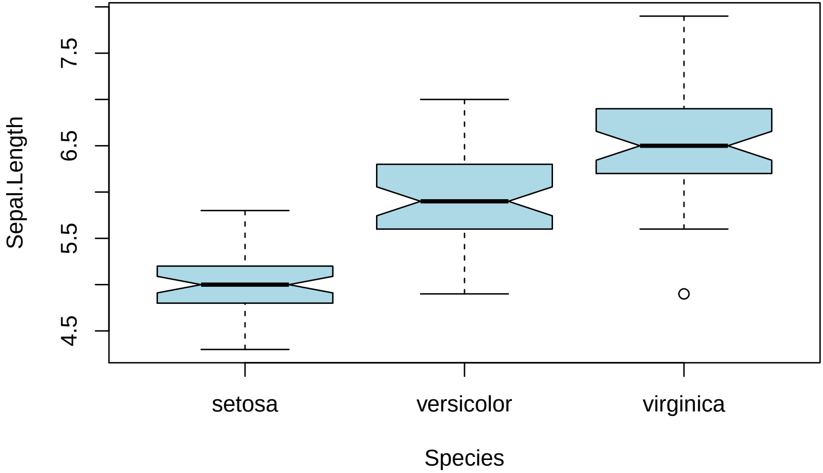
Boxplots

```r
boxplot(iris[,1:4], col='lightblue')
```



Boxplots

```
boxplot(Sepal.Length~Species, data=iris, notch = T,
        col='lightblue')
```

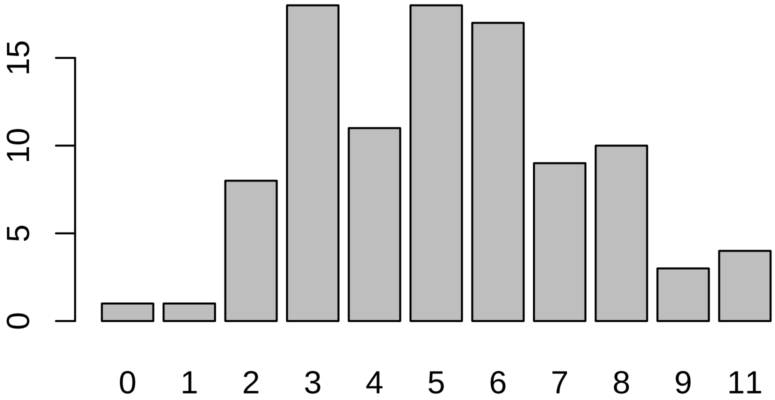


Barplots

Barplots

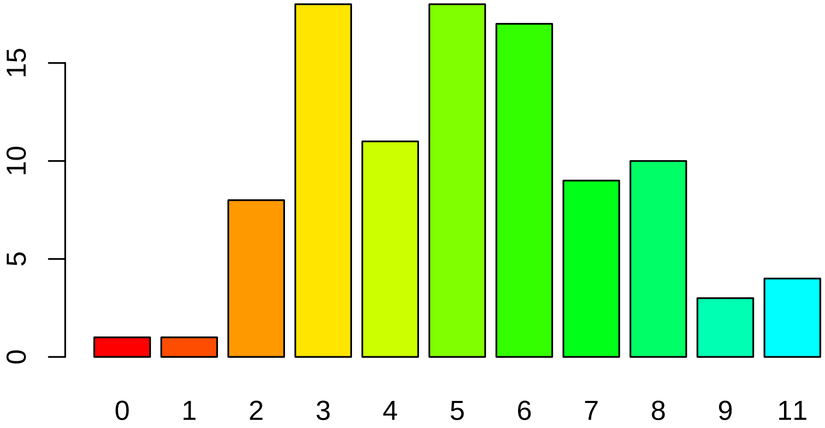
barplot draws a bar diagram for counts. We give examples combining it with table to count the number of occurrences of data values.

```
tN <- table(Ni <- rpois(100, lambda=5))
barplot(tN)
```



Barplots

```
barplot(tN, col=rainbow(20))
```



Histograms

Histograms

We will use the `morley` dataset that contains the results of (one of) Michelson's experiments on the speed of light.

Five experiments with 20 consecutive runs

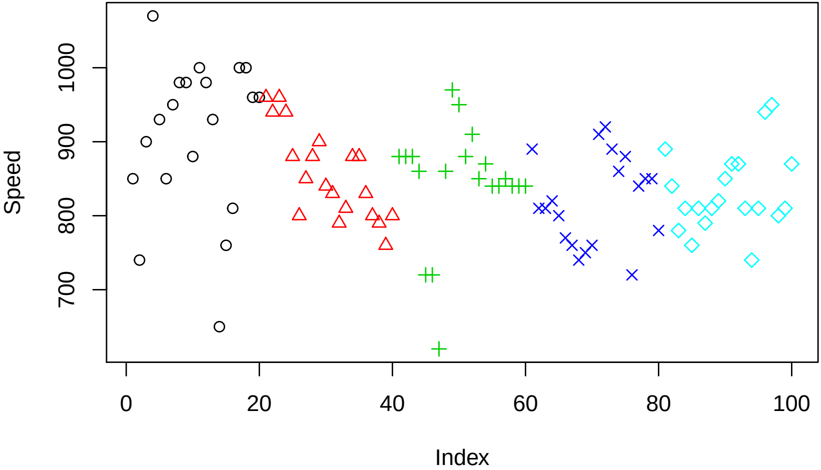
```
str(morley)
```

```
## 'data.frame':    100 obs. of  3 variables:
##  $ Expt : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ Run  : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Speed: int  850 740 900 1070 930 850 950 980 980 880
```

```
attach(morley)
```

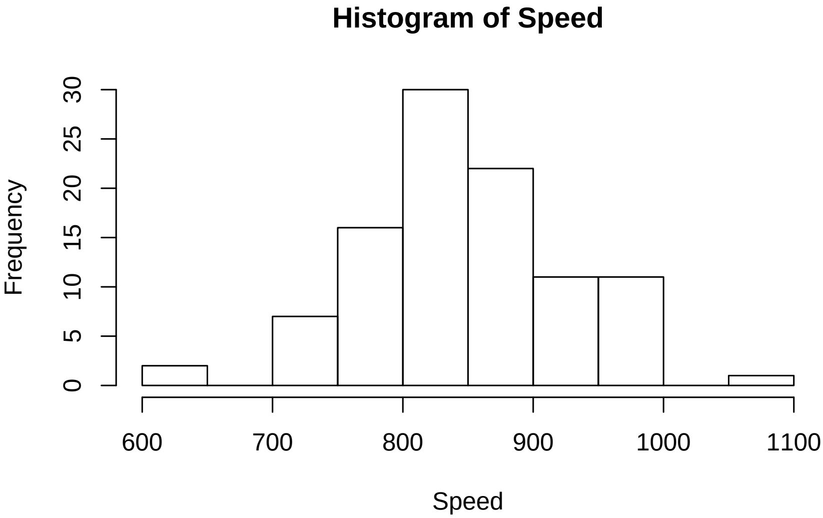
Histograms

```r
plot(Speed,col=Expt, pch=Expt)
```



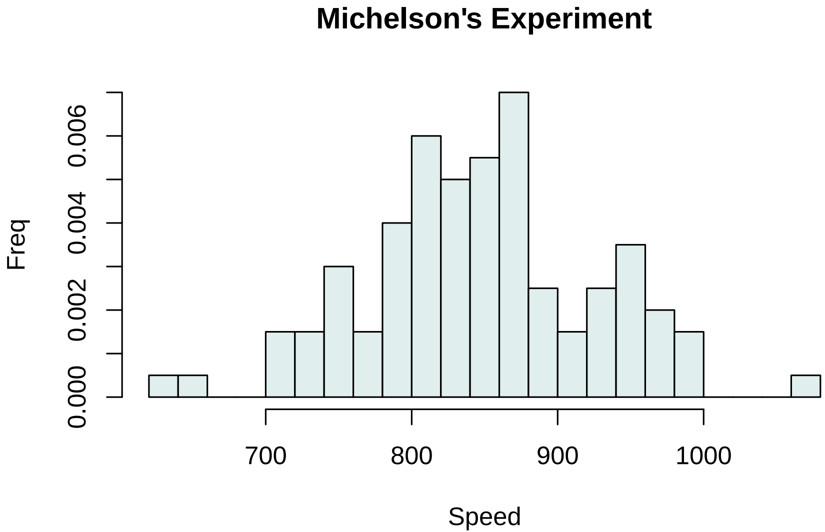
Histograms

```
hist(Speed)
```

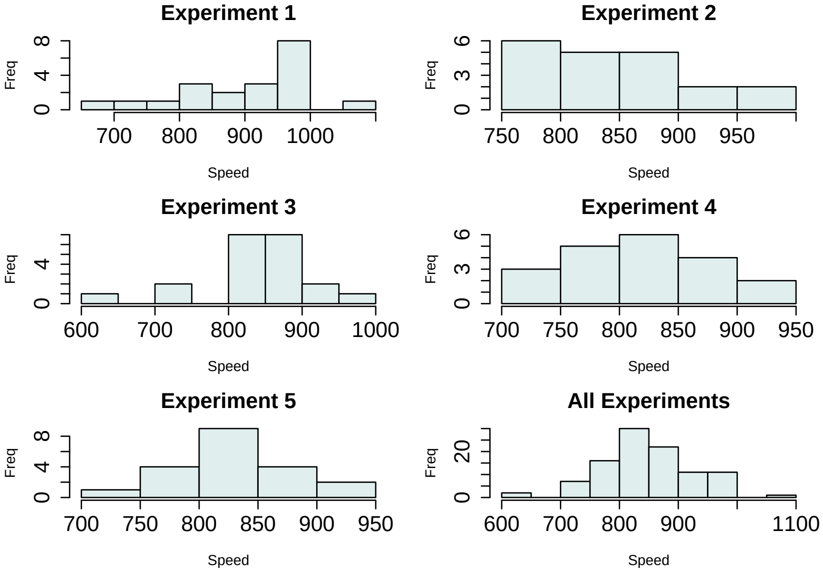


Histograms

```
hist(Speed, breaks = 20, probability = T,
     col = 'azure2',xlab='Speed', ylab='Freq',
     main = "Michelson's Experiment")
```

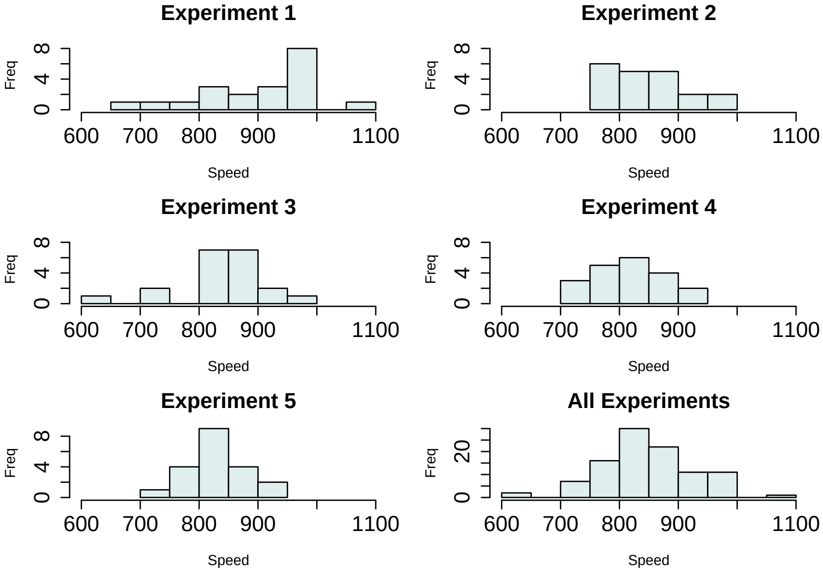


Histograms



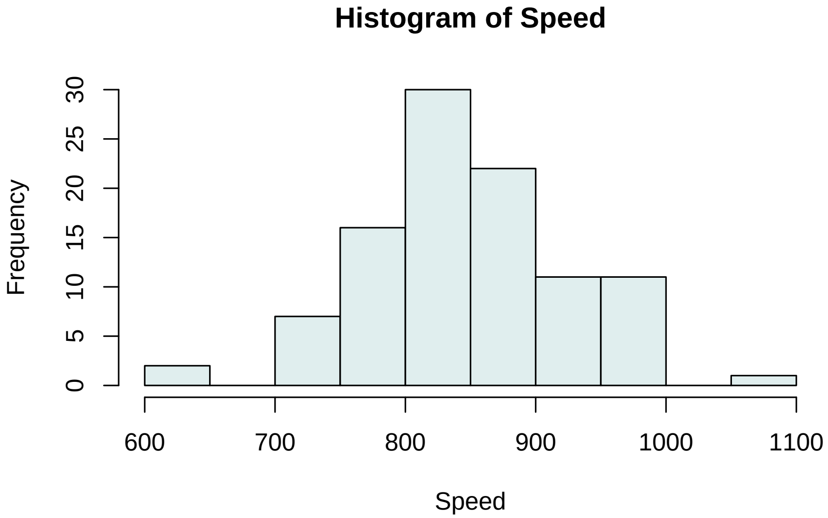
Experiment 1
Freq
0
4
8
700
800
900
1000
Speed
Experiment 2
Freq
0
3
6
750
800
850
900
950
Speed
Experiment 3
Freq
0
4
600
700
800
900
1000
Speed
Experiment 4
Freq
0
3
6
700
750
800
850
900
950
Speed
Experiment 5
Freq
0
4
8
700
750
800
850
900
950
Speed
All Experiments
Freq
0
20
600
700
800
900
1100
Speed

Histograms


Experiment 1
Experiment 2
Experiment 3
Experiment 4
Experiment 5
All Experiments
Freq
0
4
8
20
600
700
800
900
1100
Speed

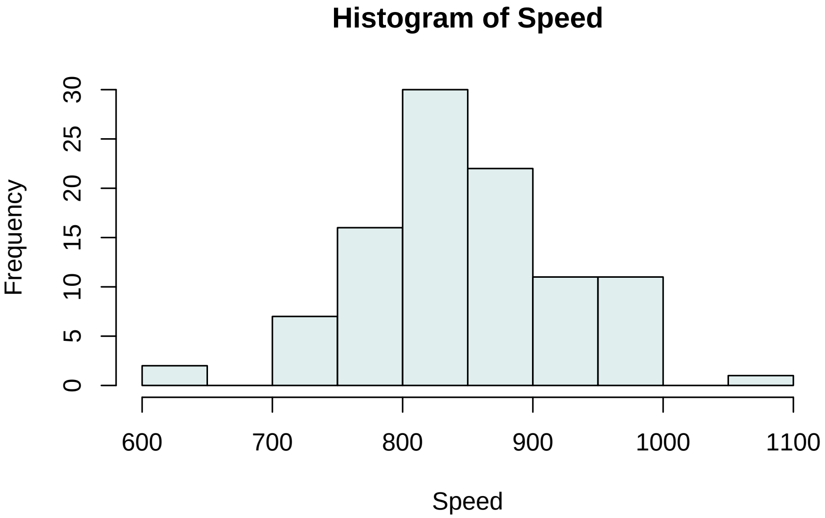
Histograms: Number of breaks

```r
hist(Speed,col = 'azure2')
```



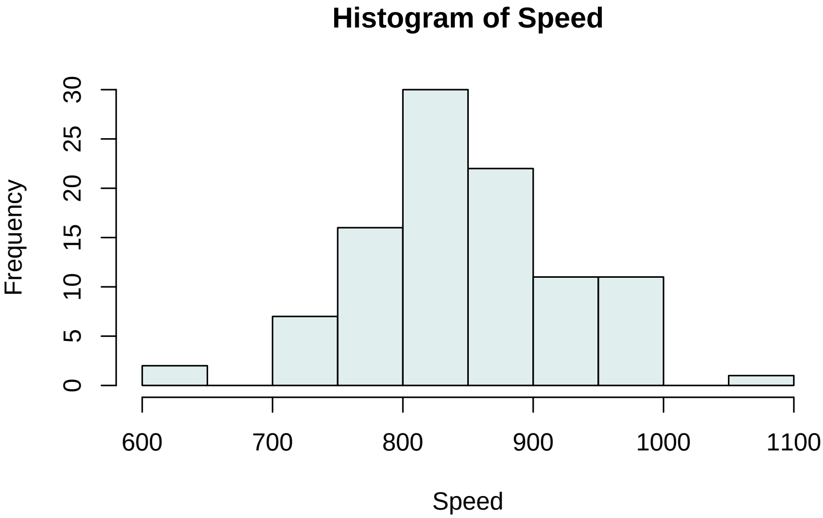
Histograms: Number of breaks

```r
hist(Speed, breaks = 8,col = 'azure2')
```

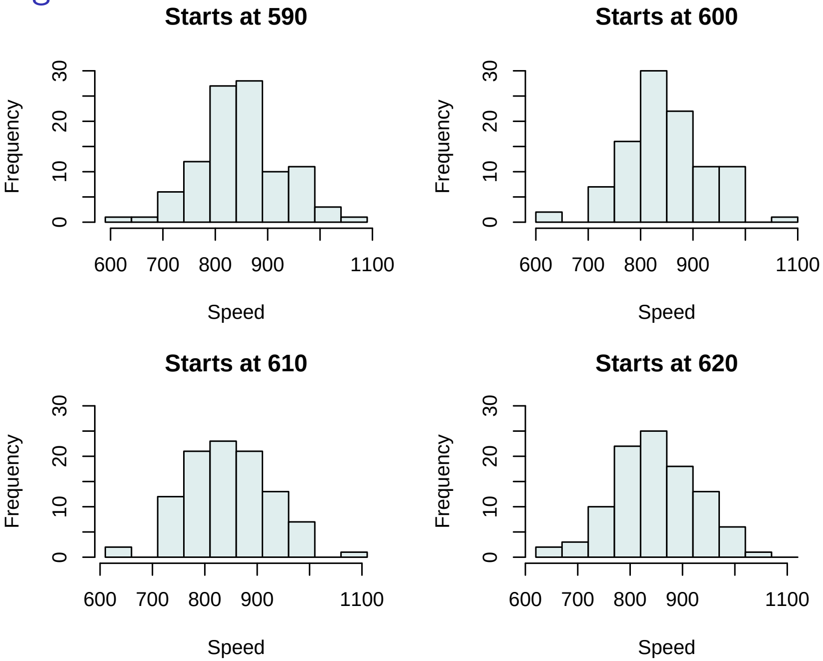


Histograms: Number of breaks

```r
hist(Speed, breaks = 16,col = 'azure2')
```

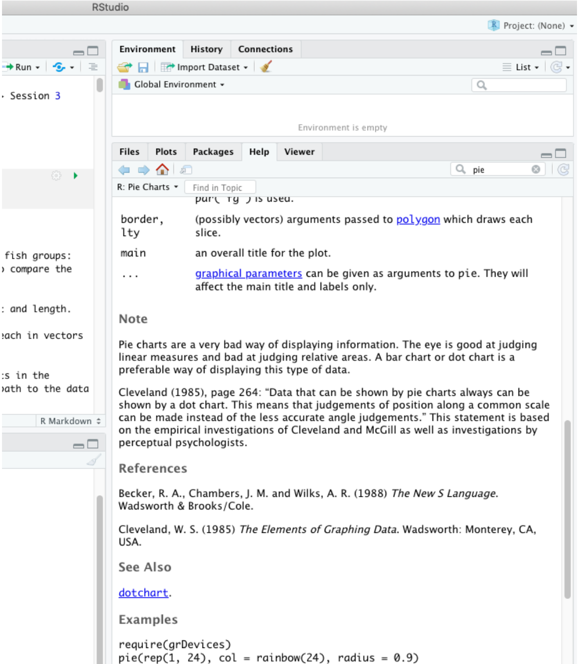


Histograms: Anchor



Dot Charts

Dot Charts



```
· Session 3

 fish groups:
› compare the

: and length.

each in vectors

:s in the
path to the data
```

par(fg) is used.

border, lty (possibly vectors) arguments passed to polygon which draws each slice.

main an overall title for the plot.

... graphical parameters can be given as arguments to pie. They will affect the main title and labels only.

Note

Pie charts are a very bad way of displaying information. The eye is good at judging linear measures and bad at judging relative areas. A bar chart or dot chart is a preferable way of displaying this type of data.

Cleveland (1985), page 264: "Data that can be shown by pie charts always can be shown by a dot chart. This means that judgements of position along a common scale can be made instead of the less accurate angle judgements." This statement is based on the empirical investigations of Cleveland and McGill as well as investigations by perceptual psychologists.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole.

Cleveland, W. S. (1985) *The Elements of Graphing Data*. Wadsworth: Monterey, CA, USA.

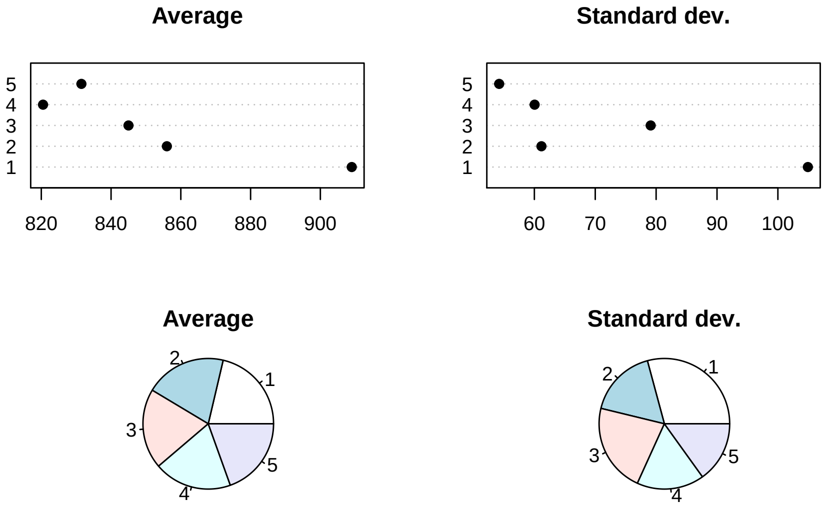
See Also

dotchart.

Examples

```
require(grDevices)
pie(rep(1, 24), col = rainbow(24), radius = 0.9)
```

Dot Charts


Average
5
4
3
2
1
820
840
860
880
900
Standard dev.
5
4
3
2
1
60
70
80
90
100
Average
1
2
3
4
5
Standard dev.
1
2
3
4
5

Dot Charts

The following pie charts present three examples where the values to be plotted are close to each other. Try to determine, in each case, the correct order.

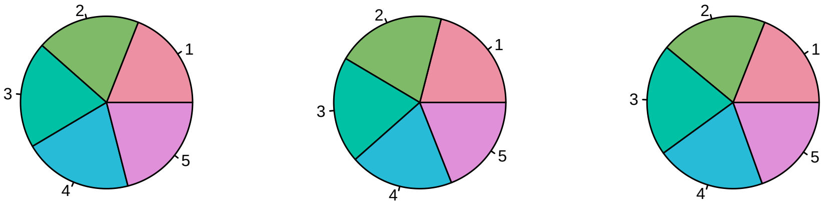


Figure 1: Pie charts for three data sets

Dot Charts

It is very difficult to determine relative sizes in a pie chart. On the other hand, if we draw a barplot,

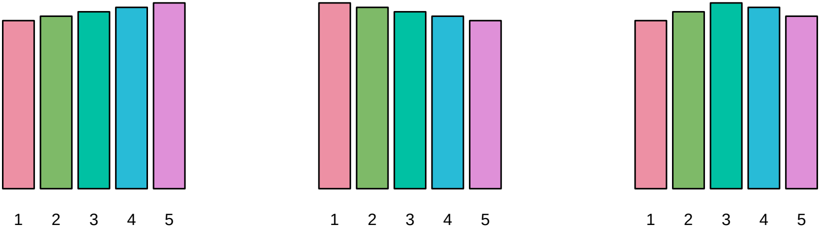


Figure 2: Bar charts for three data sets

the order is very clear in all cases.

Dot Charts

Dotcharts are also a good choice in this situation.

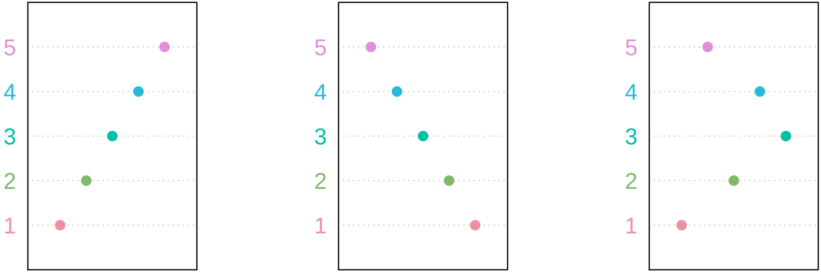


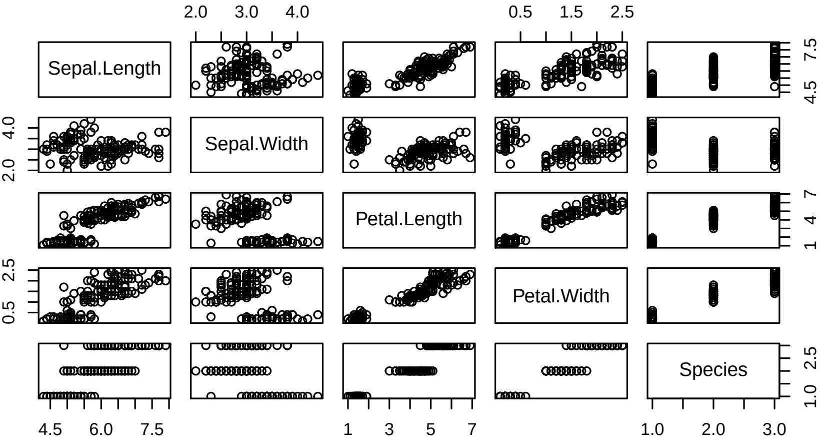
Figure 3: Dotcharts for three data sets

pairs

pairs

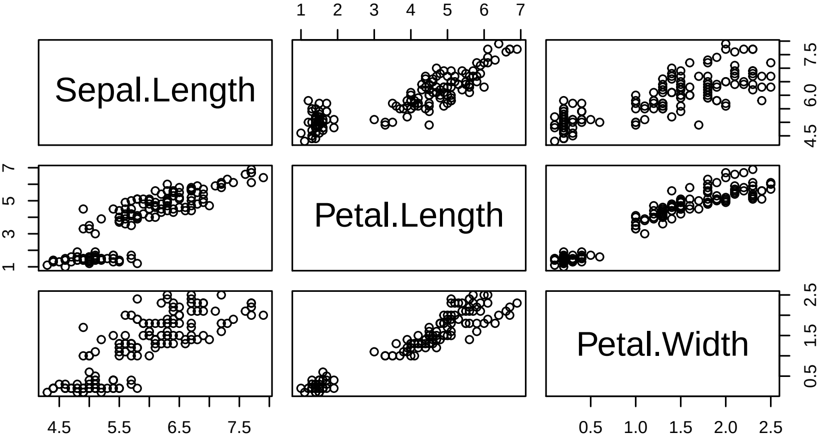
This produces a matrix of graphs:

```
pairs(iris)
```



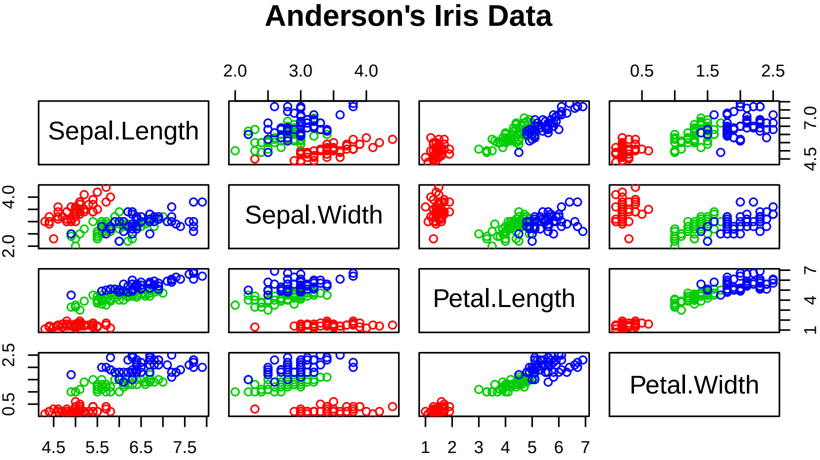
pairs

```
pairs( ~ Sepal.Length + Petal.Length +
     Petal.Width, data = iris)
```



pairs

```r
pairs(iris[1:4], main = "Anderson's Iris Data",pch = 21,
  col = c('red','green3','blue')[iris$Species])
```



contour

contour

Creates a contour plot or adds contour lines to an existing plot.

Syntax:

```r
contour(x = seq(0, 1, length.out = nrow(z)),
        y = seq(0, 1, length.out = ncol(z)),
        z, nlevels = 10, add = FALSE)
```

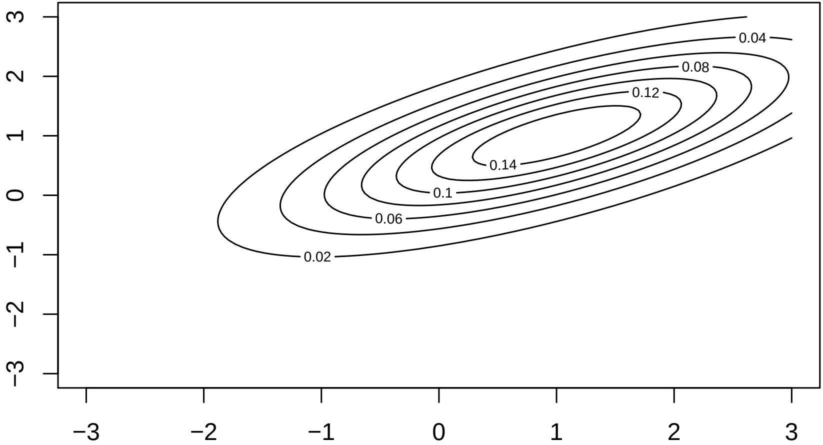
contour

Bivariate normal distribution.

```r
library(mvtnorm)
x.points <- y.points <- seq(-3,3,length.out=100)
z <- matrix(0,nrow=100,ncol=100)
mu <- c(1,1)
sigma <- matrix(c(2,1,1,1),nrow=2)
for (i in 1:100) {
  for (j in 1:100) {
  z[i,j] <- dmvnorm(c(x.points[i],y.points[j]),
                    mean=mu,sigma=sigma)
  }
}
contour(x.points,y.points,z)
```

contour

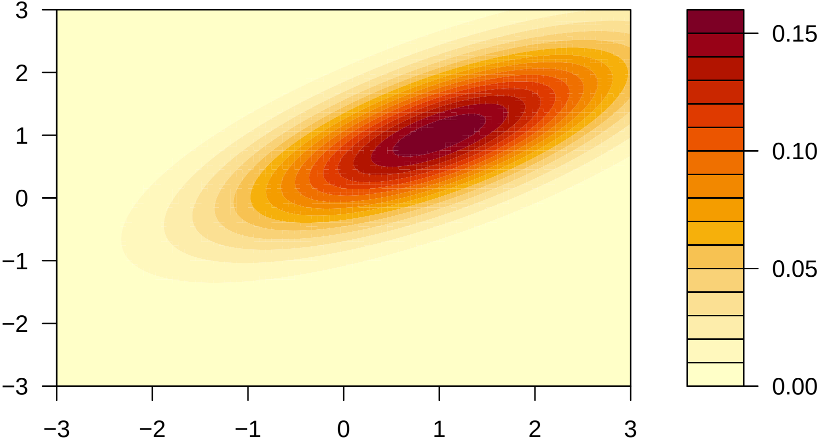
Bivariate normal distribution.



contour

Bivariate normal distribution.

```
filled.contour(x.points,y.points,z)
```



contour

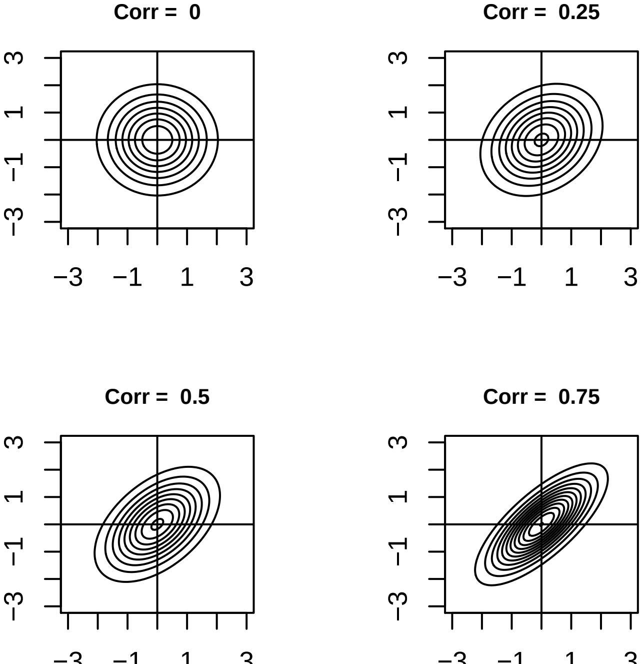
```
normal.contour <- function(mu=c(0,0),
          sigma=matrix(rep(1,4),nrow=2)){
  x.points <- y.points <- seq(-3,3,length.out=100)
  z <- matrix(0,nrow=100,ncol=100)
  for (i in 1:100) {
    for (j in 1:100) {
      z[i,j] <- dmvnorm(c(x.points[i],y.points[j]),
                        mean=mu,sigma=sigma)
    }
  }
  contour(x.points,y.points,z, main=paste('Corr = ',
        sigma[1,2]), cex.main = 0.8, drawlabels = FALSE)
  abline(h=0); abline(v=0)
}
```

contour

```r
mu = c(0,0)
sigma1 <- matrix(c(1,0,0,1),nrow=2)
sigma2 <- matrix(c(1,.25,.25,1),nrow=2)
sigma3 <- matrix(c(1,.5,.5,1),nrow=2)
sigma4 <- matrix(c(1,.75,.75,1),nrow=2)

normal.contour(mu,sigma1)
normal.contour(mu,sigma2)
normal.contour(mu,sigma3)
normal.contour(mu,sigma4)
```

contour



Corr = 0

Corr = 0.25

Corr = 0.5

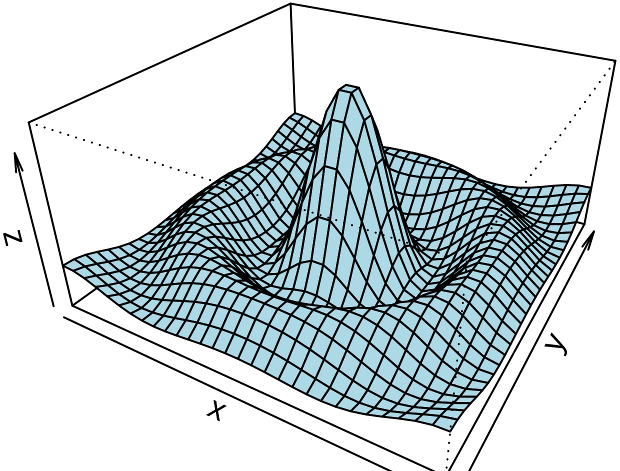
Corr = 0.75

persp

persp

This function displays a perspective plot of a surface over the *xy* plane.

```
y <- x <- seq(-10, 10, length= 30)
f <- function(x, y) { r <- sqrt(x^2+y^2); 10 * sin(r)/r }
z <- outer(x, y, f); z[is.na(z)] <- 1
persp(x, y, z, theta = 30, phi = 30, expand = 0.5, col = "lightblue")
```



Other Chart Types

Other Chart Types

```
sunflowerplot(x,y)

stripchart(x)

matplot(x,y)

plot.ts(x)

image(x,y,z)

stars(x)
```