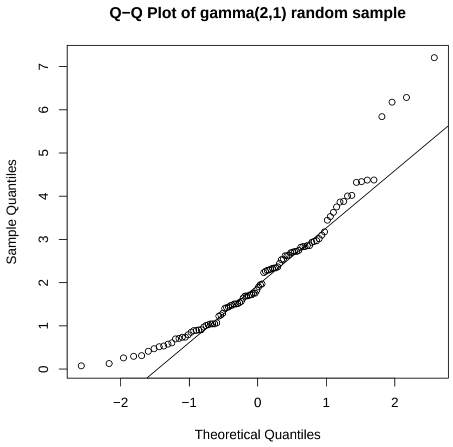
More High-Level Plots: `qqnorm`

Parameters:

- `plot` parameters
- `datax=T`: put data on x-axis instead

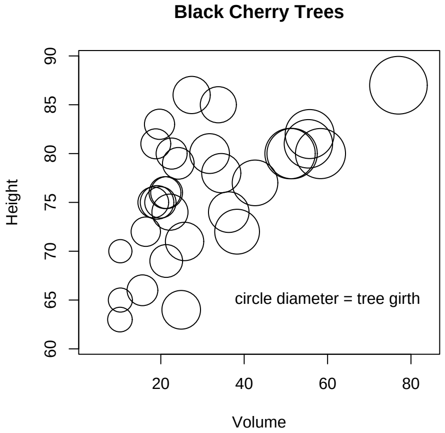
```
> r <- rgamma(100,2,1)
> qqnorm(r, main="Q-Q Plot of gamma(2,1) random sample")
> qqline(r)
>
```



Multivariate Data: `symbols`

Plots (variable-size) circles, squares, rectangles, stars, thermometers, and boxplots.

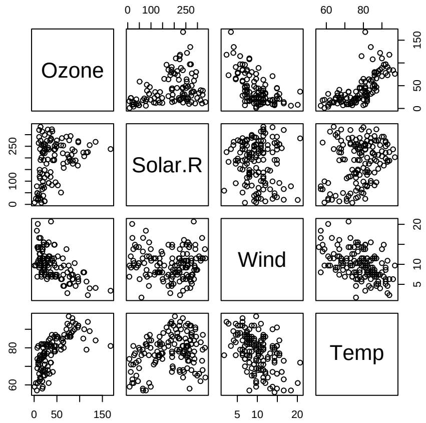
```
> trees
   Girth Height Volume
1    8.3     70   10.3
2    8.6     65   10.3
[ . . . ]
31  20.6     87   77.0
> symbols(Volume, Height, circles=Girth, inches=.3,
          main="Black Cherry Trees")
> text(60,65,"circle diameter = tree girth")
>
```



Multivariate Data: `pairs`

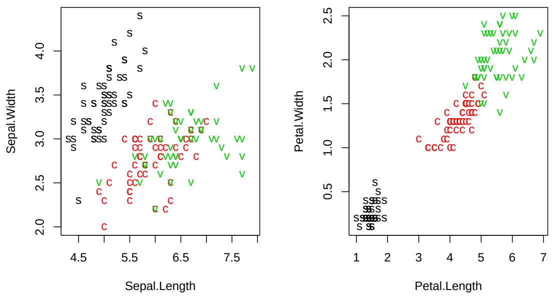
Produces scatterplots of all pairs of columns in a matrix. Good way to quickly gauge relationships between several continuous variables.

```
> airquality  # some New York air quality measurements
    Ozone Solar.R Wind Temp Month Day
1      41     190  7.4   67     5   1
2      36     118  8.0   72     5   2
[ . . . ]
> attach(airquality)
> pairs(cbind(Ozone,Solar.R,Wind,Temp))
>
```



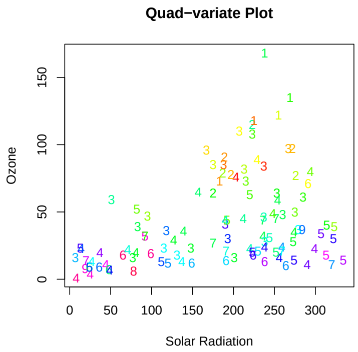
Multivariate Data: `col` and `pch`

```
> iris
  Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1          5.1         3.5          1.4         0.2     setosa
2          4.9         3.0          1.4         0.2     setosa
> attach(iris)
> levels(Species)
[1] "setosa"      "versicolor" "virginica"
> as.numeric(Species)
  [1] 1 1 1 [ . . . ] 3 3 3
> plot(Sepal.Length, Sepal.Width, col=as.numeric(Species),
       pch=c("s","c","v")[as.numeric(Species)])
> plot(Petal.Length, Petal.Width, col=as.numeric(Species),
       pch=c("s","c","v")[as.numeric(Species)])
>
```



Multivariate Data: `col` and `pch`

```
> attach(airquality)
> names(airquality)
[1] "Ozone"   "Solar.R" "Wind"    "Temp"    "Month"   "Day"
> range(Wind)
[1]  1.7 20.7
> range(Temp)
[1] 56 97
> Wind.char <- as.character(floor(9.9*Wind/max(Wind)))
> Wind.char
  [1] "3" "3" "6" "5" "6" "7" "4" "6" "9" "4" "3" "4" "4" [ . . . ]
> Temp.col <- rev(rainbow(diff(range(Temp))+1))[Temp-min(Temp)+1]
> Temp.col
  [1] "#4900FF" "#006DFF" "#00B6FF" "#FF00FF" "#FF0024" [ . . . ]
> plot(Solar.R, Ozone, pch=Wind.char, col=Temp.col,
      main="Quad-variate Plot",
      xlab="Solar Radiation", ylab="Ozone")
>
```

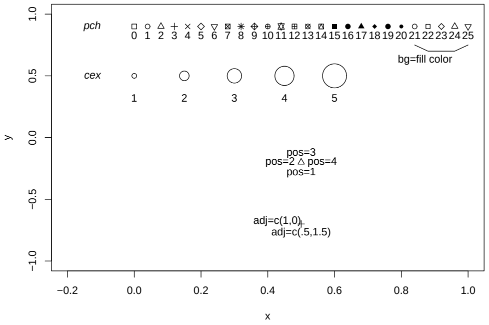


`points`, `text`, and `identify`

- `points(`*x*`,` *y*`, pch=`*plot chars*`)`
  Adds points `(x[1],y[1])`, . . . , `(x[`n`],x[`n`])`. The `pch` vector is *recycled*: it can specify a symbol number 0–25 or a single character.
  Some other parameters:
  - `type`: as for `plot(...)`
  - `col`: symbol color (recycled)
  - `bg`: fill color for `pch=21:25` (recycled)
  - `cex`: symbol scale (recycled)
- `text(`*x*`,` *y*`,` *labels*`)`
  Adds labels to given points.
  Some other parameters:
  - `pos=1,2,3,4` position below, left, above, or right
  - `offset=0.5` offset for `pos`
  - `adj=c(0.5,0.5)` alternative to `pos`
  - `col`, `cex`: as above
  - `font=1,2,3,4` for normal, bold, italic, bold italic
- `identify(`*x*`,` *y*`,` *labels*`)`
  Like `text`, but add labels interactively with left mouse button and finish with right button. Also, default labels are 1, 2, . . .

`points` and `text`

```
> plot(c(-.2,1),c(-1,1),type="n",xlab="x",ylab="y")
> x <- seq(from=0,to=1,length=26)
> text(-.1, .9, "pch", adj=1, font=3)
> points(x, rep(.9,26), pch=0:25)
> text(x, rep(.9,26), 0:25, pos=1)
> lines(x[c(22,23,25,26)],c(.75,.7,.7,.75))
> text(x[24],.7,"bg=fill color",adj=c(.8,1.5))
> x <- seq(from=0,to=.6,length=5)
> text(-.1, .5, "cex", adj=1, font=3)
> points(x, rep(.5,5), pch=1, cex=1:5)
> text(x, rep(.5,5), 1:5, pos=1, offset=1.5)
> points(.5,-.2,pch=2)
> text(.5,-.2,paste("pos=",1:4,sep=""),pos=1:4)
> points(.5,-.7,pch=3)
> text(.5,-.7,"adj=c(1,0)",adj=c(1,0))
> text(.5,-.7,"adj=c(.5,1.5)",adj=c(.5,1.5))
```

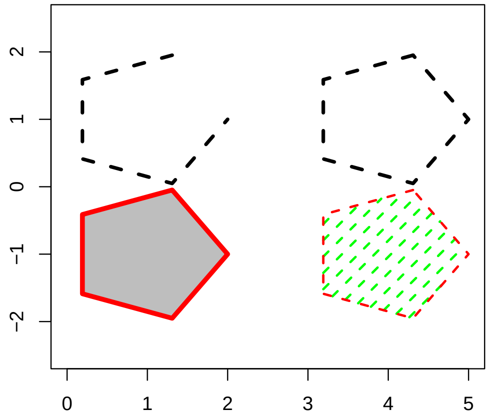


`lines` and `polygon`

- `lines(`*x*`,` *y*`)`
  Draws lines connecting points `(x[1],y[1])`, . . . , `(x[`n`],y[`n`])`. Any `NA` coordinates add a break, making multiple lines.
  Some parameters:
  - `type`: as for `plot(...)`
  - `col`: line color
  - `lty`: line type
  - `lwd`: line width
- `polygon(`*x*`,` *y*`)`
  Creates a polygon with given vertices. Any `NA` coords separate multiple polygons.
  Some parameters:
  - `col`: fill color
  - `density`, `angle`: hash shading
  - `border`: border color
  - `lty`, `lwd`: for border (and hash shading)

lines and polygon

```
> plot(0,0, type="n", xlim=c(0,5),ylim=c(-2.5,2.5),
       xlab="", ylab="")
> x <- cos(seq(from=0,to=2*pi,length=6)[-1])
> y <- sin(seq(from=0,to=2*pi,length=6)[-1])
> lines(1+x,1+y, lwd=3, lty="dashed")
> polygon(4+x,1+y, lwd=3, lty="dashed")
> polygon(1+x,-1+y, col="grey", border="red", lwd=4)
> polygon(4+x,-1+y, col="green", density=10, border="red",
          lty="dashed", lwd=2)
```



segments, arrows, and rect

All of form: `segments(`x_1`,`y_1`,`x_2`,`y_2`)`

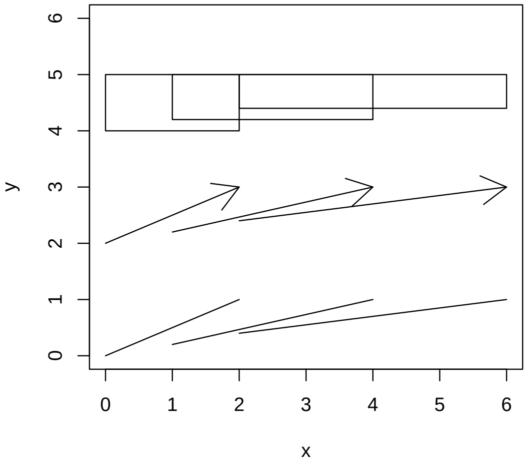
For the pairs of points

$$\begin{array}{ccc}
(x_1\texttt{[1]},y_1\texttt{[1]}) & \text{and} & (x_2\texttt{[1]},y_2\texttt{[1]}) \\
(x_1\texttt{[2]},y_1\texttt{[2]}) & \text{and} & (x_2\texttt{[2]},y_2\texttt{[2]}) \\
 & \vdots & \\
(x_1\texttt{[n]},y_1\texttt{[n]}) & \text{and} & (x_2\texttt{[n]},y_2\texttt{[n]})
\end{array}$$

- `segments` draws `n` line segments;
- `arrows` draws `n` arrows (heads at second point);
- `rect` draws `n` rectangles with given diagonally opposite points.

segments, arrows, and rect

```
> plot(c(0,6),c(0,6),type="n",xlab="x",ylab="y")
> x1 <- c(0,1,2); y1 <- c(0,.2,.4)
> x2 <- c(2,4,6); y2 <- c(1,1,1)
> segments(x1,y1,x2,y2)
> arrows(x1,2+y1,x2,2+y2)
> rect(x1,4+y1,x2,4+y2)
```



Other Annotating Functions

- `abline`
  - `abline(`*intercept*`,`*slope*`)`
  - `abline(h=c(0,2,3))`: horizontal lines with given y-values
  - `abline(v=c(-1,3,4))`: vertical lines with given x-values
- `matpoints` and `matlines`

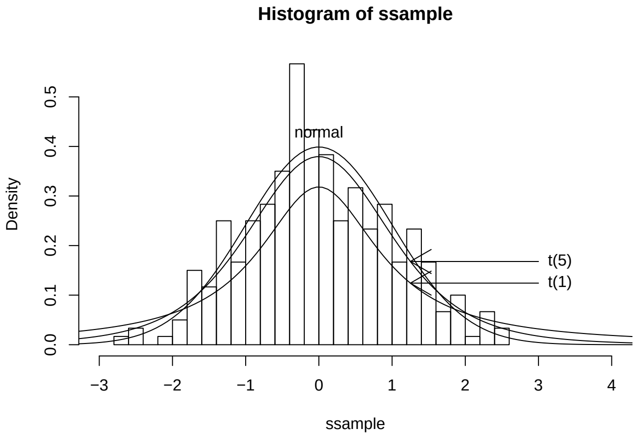
  Take matrices for x and y. Useful for plotting several columns of y-values against the same vector of x-values.
- `curve`

  Plots a function at 101 equally spaced values.

  ```
  > curve(sin(x),from=0,to=2*pi)
  > curve(cos(x),from=0,to=2*pi,add=T)
  ```

Example: Annotated Histogram

```
> ssample <- (sample-mean(sample))/sd(sample)
> hist(ssample, probability=T, nclass=25, xlim=c(-3,4))
> curve(dnorm(x), add=T)
> for (df in c(1,5))
     curve(dt(x, df), add=T)
> text(0, dnorm(0),
       "normal", pos=3)
> y <- dt(1.25, df=c(5,1))
> text(3, y, c("t(5)", "t(1)"), pos=4)
> arrows(3, y, 1.25, y)
```


Histogram of ssample
Density
normal
t(5)
t(1)
ssample