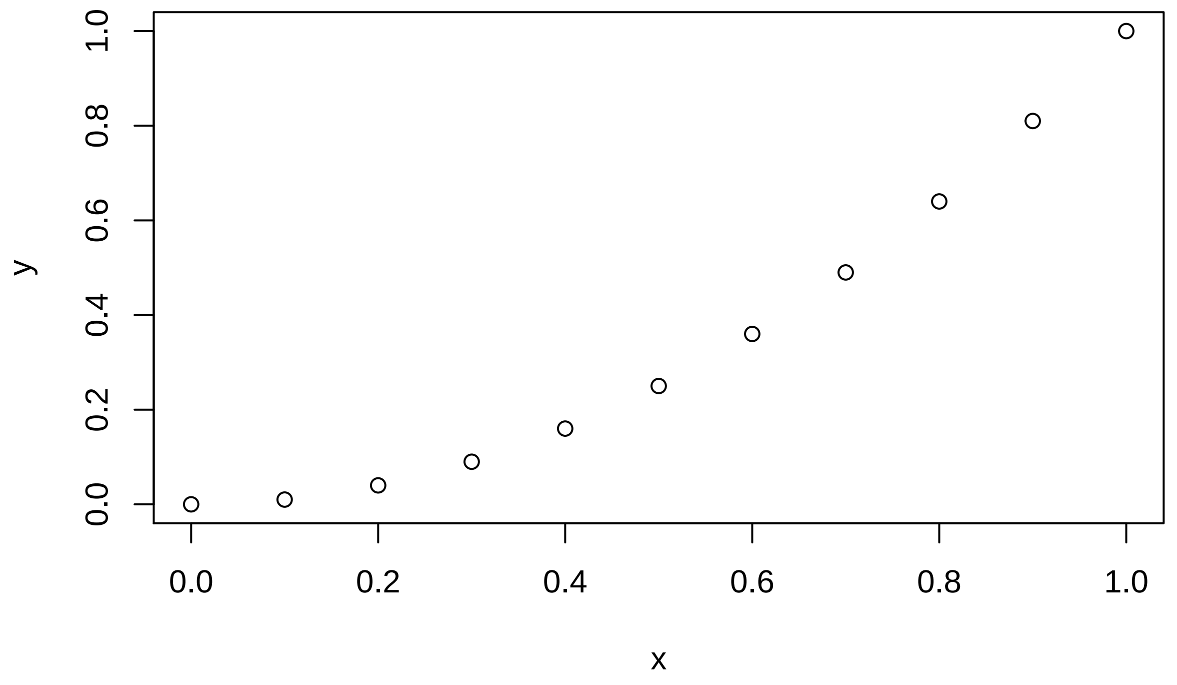
Traditional Graphics

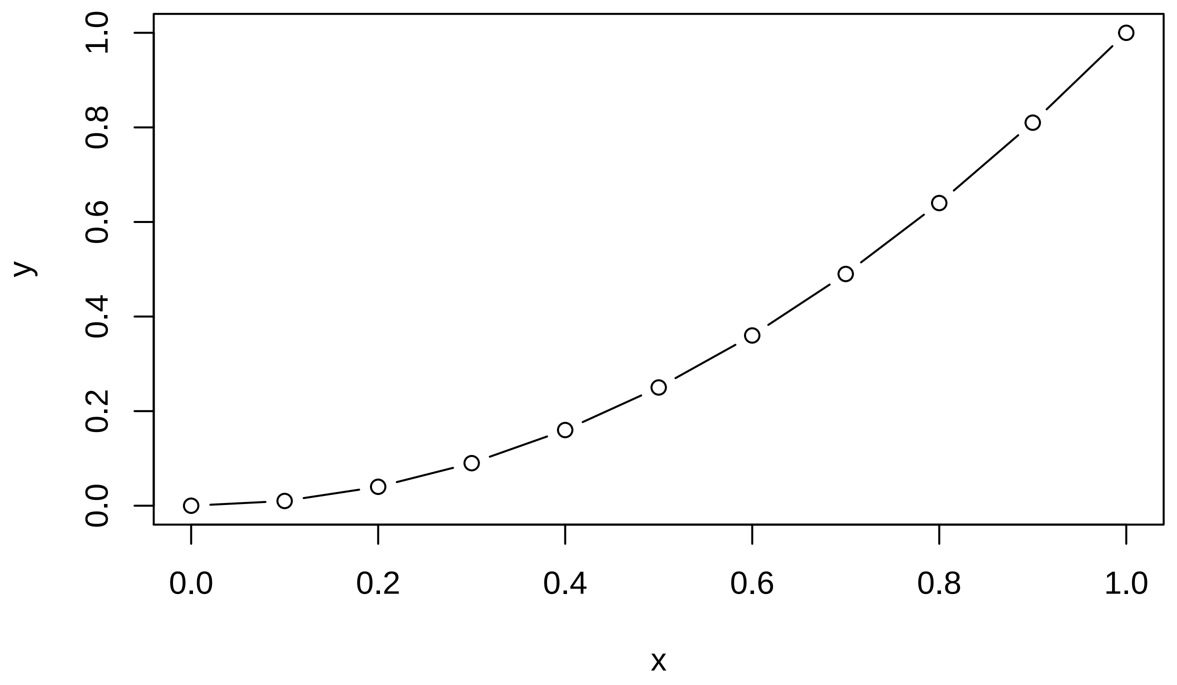
The `plot` function can be used to make 2-d plots of (discrete) data.

```r
x <- seq(0,1,0.1)
y <- x^2
plot(x, y)
```



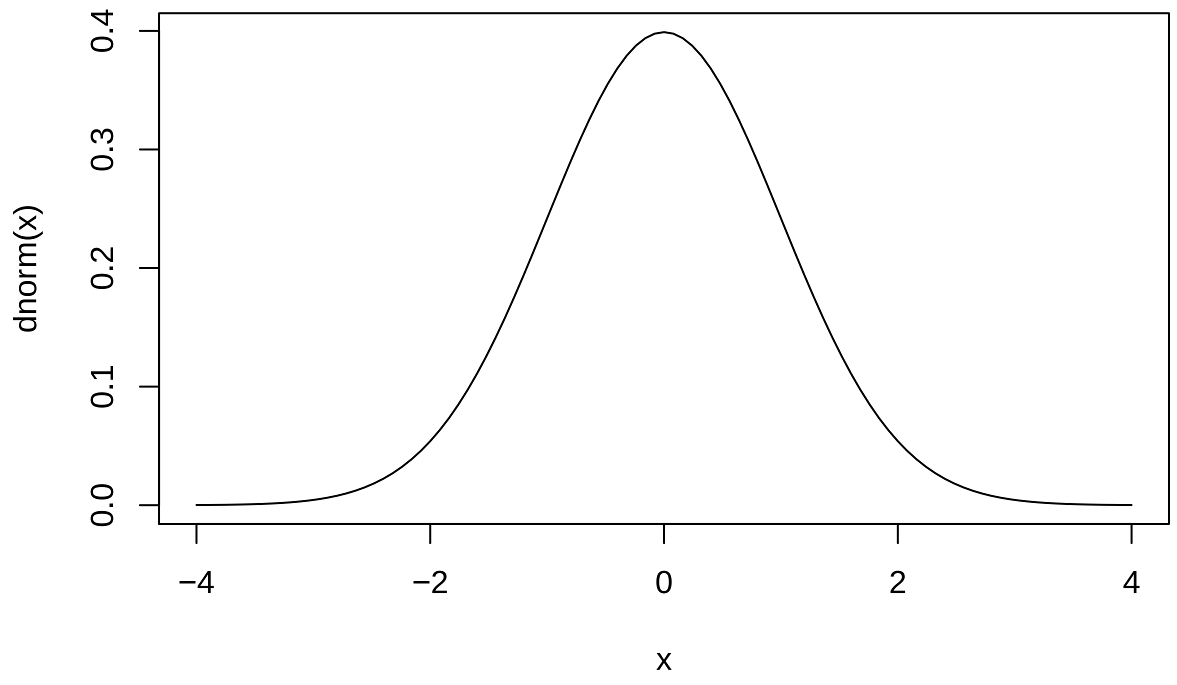
We can also connect the points with lines.

```r
x <- seq(0,1,0.1)
y <- x^2
plot(x, y, type = "b")
```



We can plot a function, say the normal density function `dnorm`.

```r
curve(dnorm, -4, 4)
```

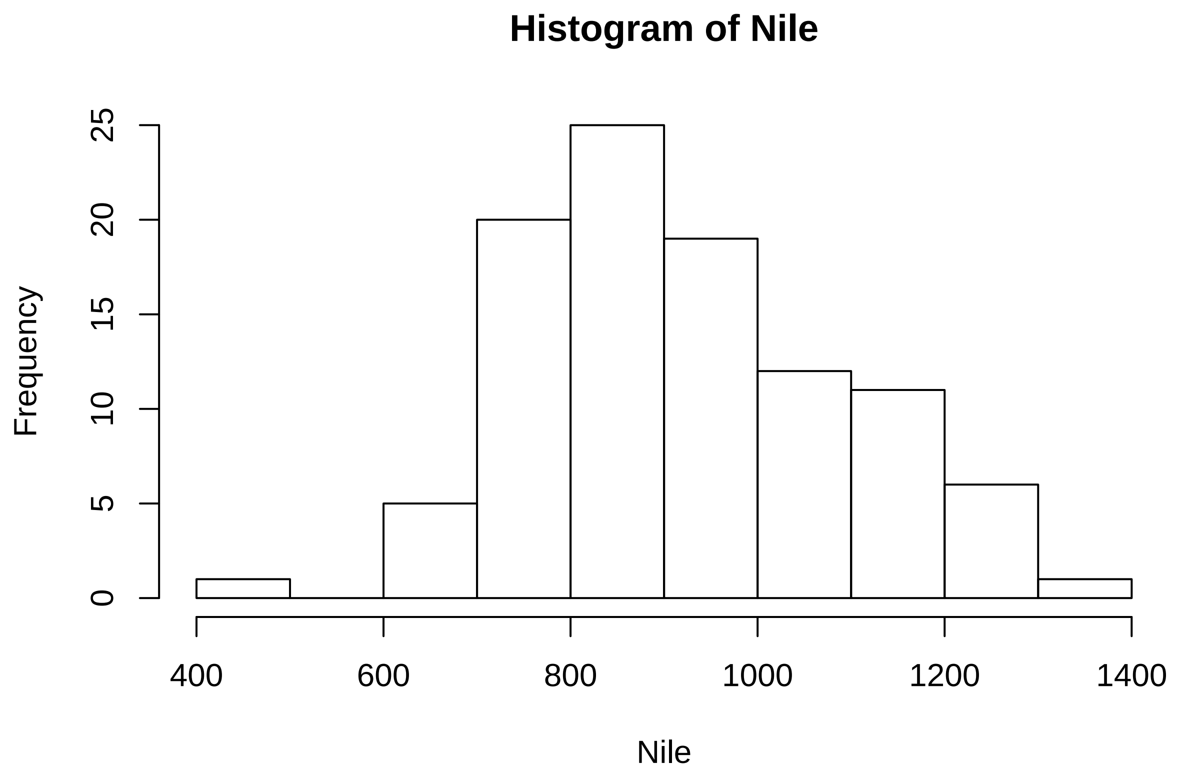


Create a histogram using the included `Nile` dataset.

```r
print(Nile)
```

```
## Time Series:
## Start = 1871
## End = 1970
## Frequency = 1
##   [1] 1120 1160  963 1210 1160 1160  813 1230 1370 1140  995  935 1110  994
##  [15] 1020  960 1180  799  958 1140 1100 1210 1150 1250 1260 1220 1030 1100
##  [29]  774  840  874  694  940  833  701  916  692 1020 1050  969  831  726
##  [43]  456  824  702 1120 1100  832  764  821  768  845  864  862  698  845
##  [57]  744  796 1040  759  781  865  845  944  984  897  822 1010  771  676
##  [71]  649  846  812  742  801 1040  860  874  848  890  744  749  838 1050
##  [85]  918  986  797  923  975  815 1020  906  901 1170  912  746  919  718
##  [99]  714  740
```

```r
hist(Nile)
```

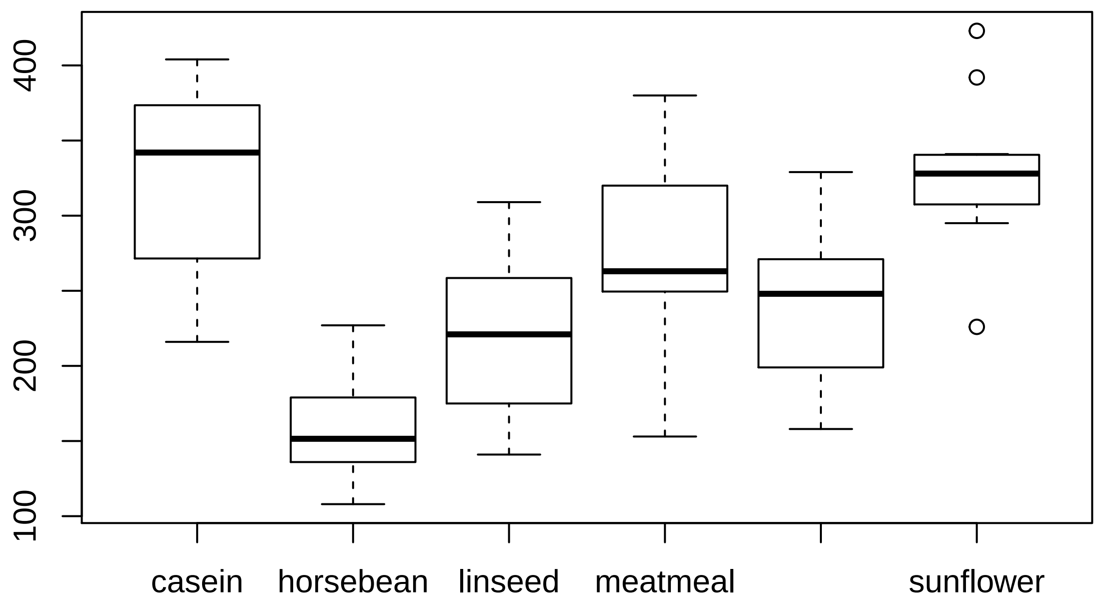


Boxplots can be created very quickly. Consider the `chickwts` dataset.

```
head(chickwts)
```

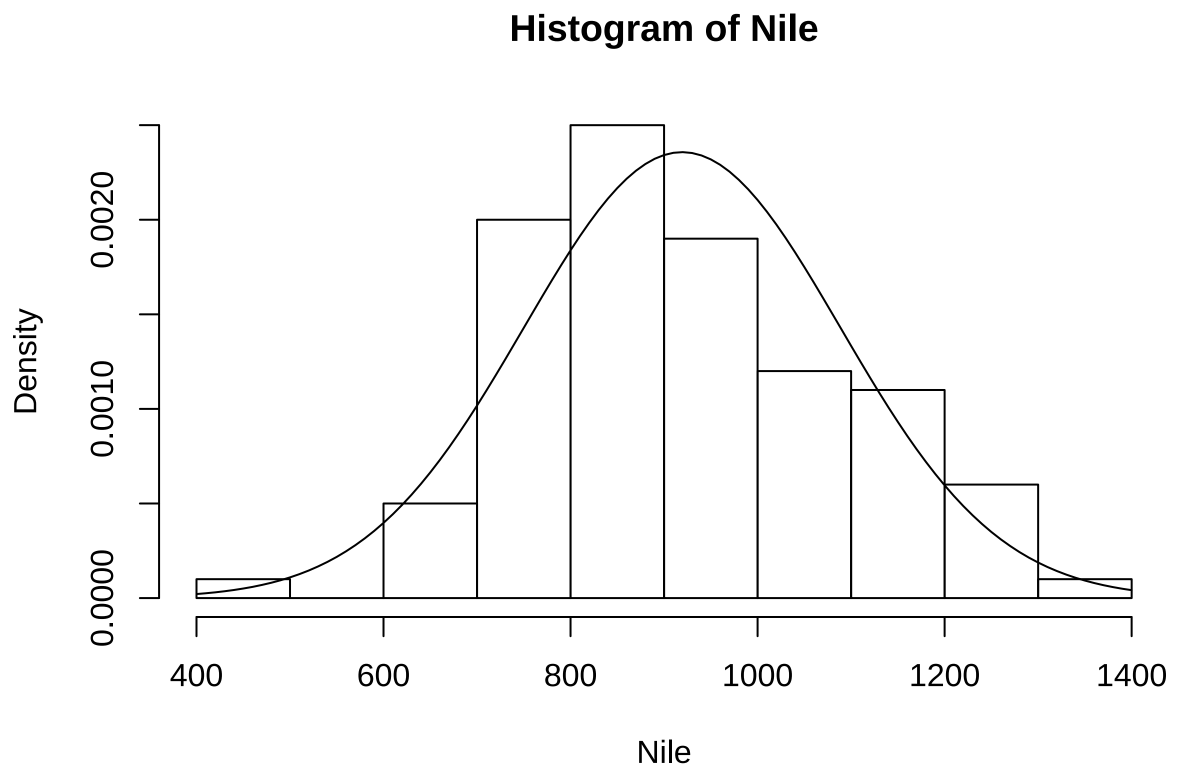
```
##   weight      feed
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## 5    217 horsebean
## 6    168 horsebean
```

```
boxplot(weight ~ feed, data = chickwts)
```



Overlay a curve on a plot.

```
mu.hat <- mean(Nile)
sigma.hat <- sd(Nile)
hist(Nile, freq = FALSE)
curve(dnorm(x, mean = mu.hat, sd = sigma.hat), add = TRUE)
```



Instead of having the plot display in a window, request it to be written to a file using functions like `pdf`, `jpeg`, and `bmp`.

```
pdf("plot.pdf", height=4, width=4)
hist(Nile)
dev.off()
```

- The function `dev.off()` closes the graphics file after writing to it.
- The units of `height` and `weight` are inches.
- To save a plot during Rstudio during interactive use, see the `Export` menu above the plot.

There are lower-level plotting functions available for drawing shapes. `abline` can be used to draw lines, e.g. to plot a simple linear regression.

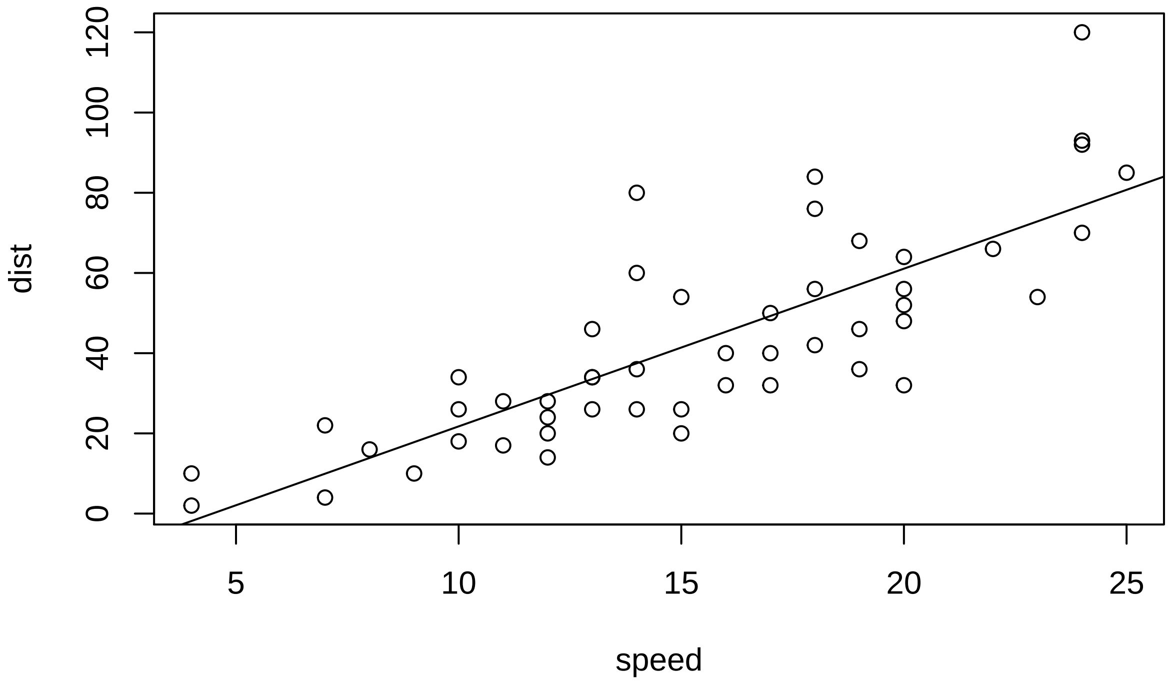
```
head(cars)
```

```
##   speed dist
## 1     4    2
## 2     4   10
## 3     7    4
## 4     7   22
## 5     8   16
## 6     9   10
```

```
fit <- lm(cars$dist ~ cars$speed)
print(fit)
```

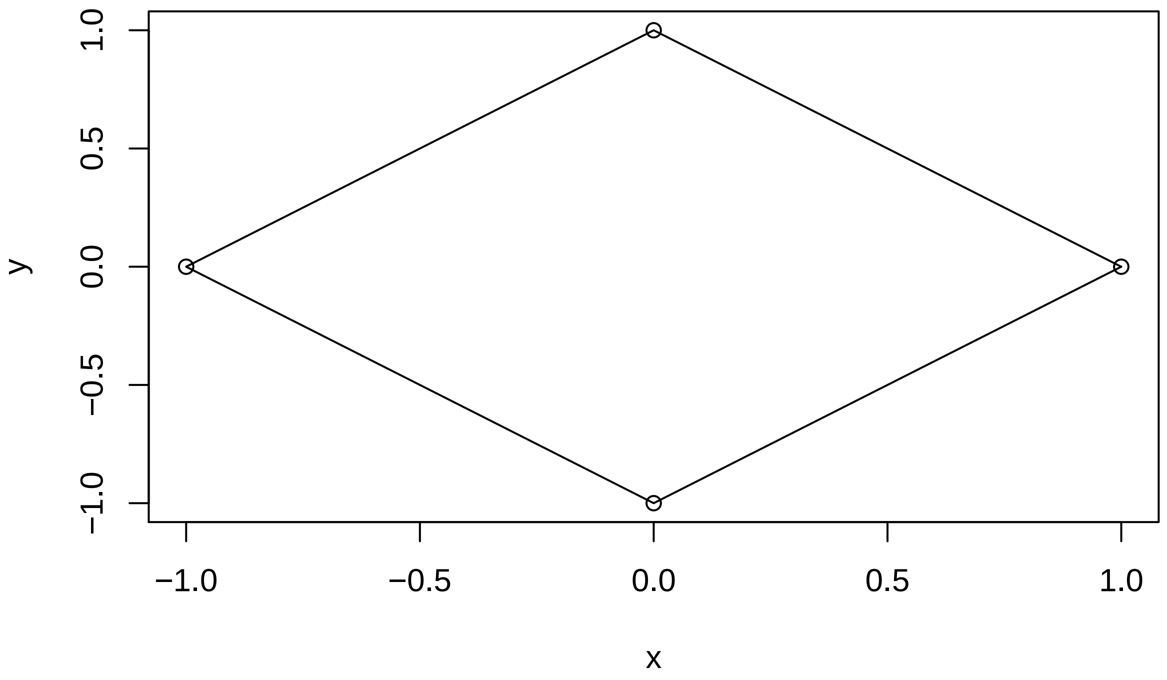
```
##
## Call:
## lm(formula = cars$dist ~ cars$speed)
##
## Coefficients:
## (Intercept)   cars$speed
##     -17.579        3.932
```

```
plot(cars)
abline(coef = fit$coefficients)
```



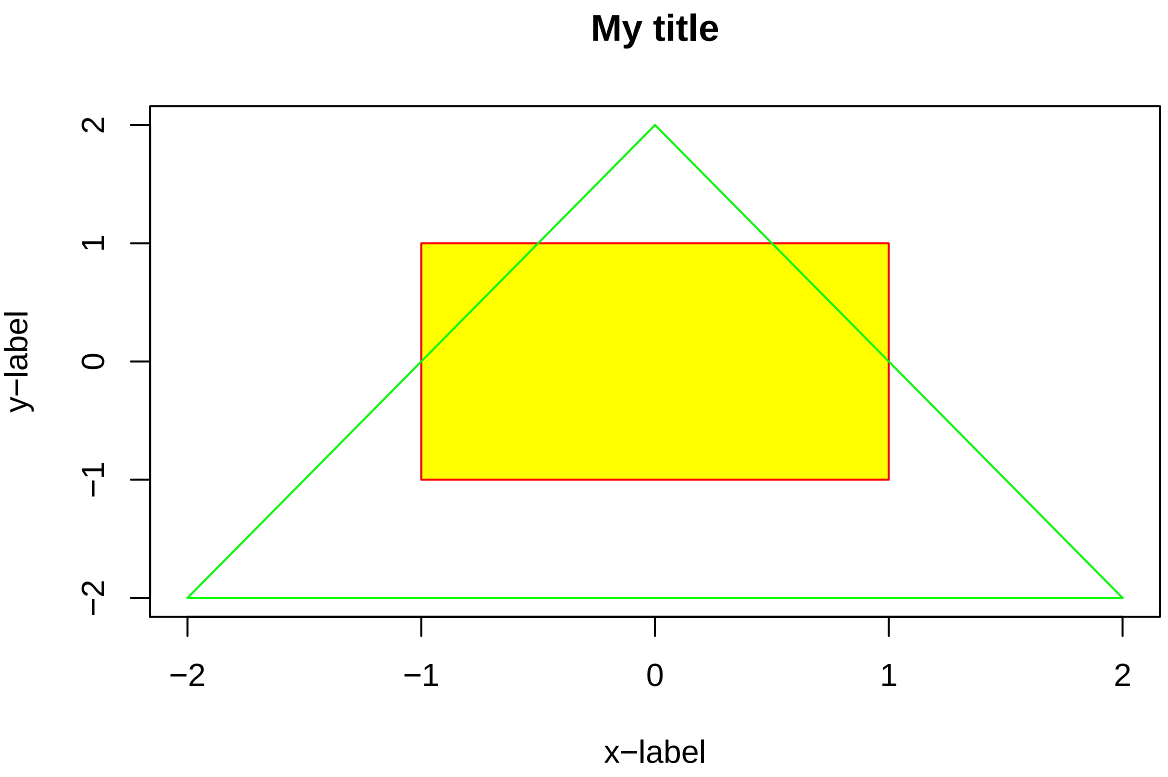
Draw polygons by connecting points.

```r
x <- c(1, 0, -1, 0)
y <- c(0, 1, 0, -1)
plot(x,y)
polygon(x,y)
```



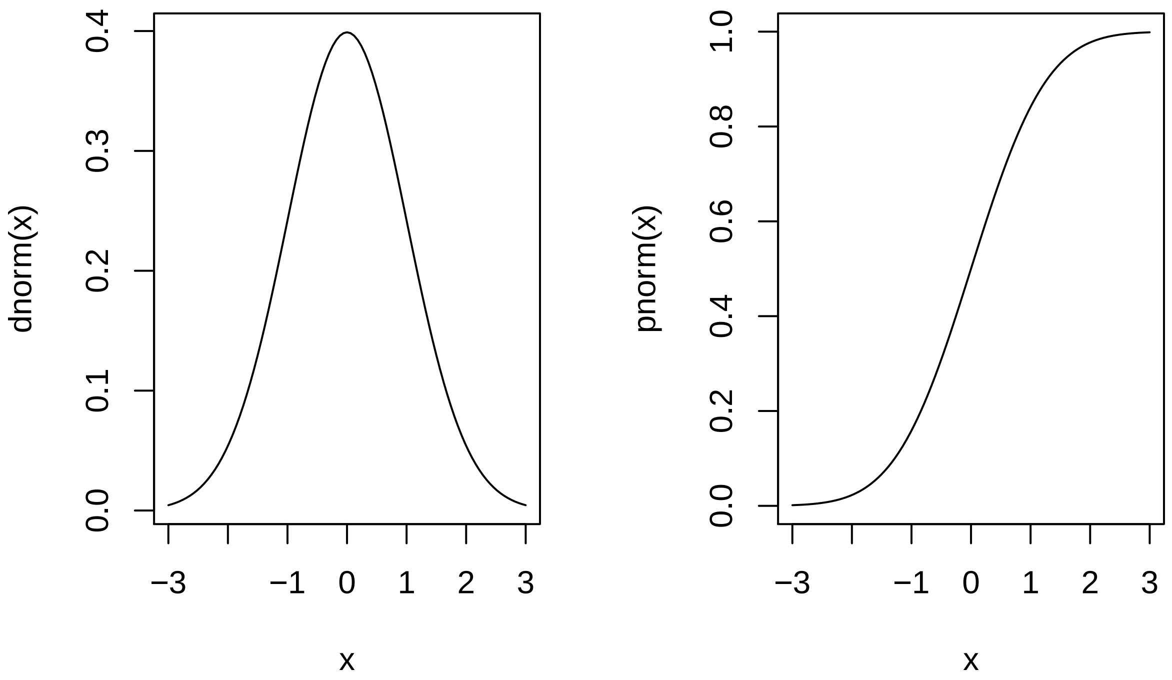
Draw rectangles with `rect` and line segments with `segments`.

```r
plot.new()  # Create an empty plot window, and add the parts manually
plot.window(xlim=c(-2,2), ylim=c(-2,2))
axis(1)
axis(2)
title(main="My title")
title(xlab="x-label")
title(ylab="y-label")
box()
rect(xleft = -1, ybottom = -1, xright = 1, ytop = 1, col="yellow", border = "red")
segments(x0 = -2, y0 = -2, x1 = 0, y1 = 2, col="green")
segments(x0 = 0, y0 = 2, x1 = 2, y1 = -2, col="green")
segments(x0 = -2, y0 = -2, x1 = 2, y1 = -2, col="green")
```



Multiple plots in one figure.

```r
par(mfrow = c(1,2))
curve(dnorm(x), xlim = c(-3,3))
curve(pnorm(x), xlim = c(-3,3))
```



```r
dev.off()
```

```
## null device
##           1
```