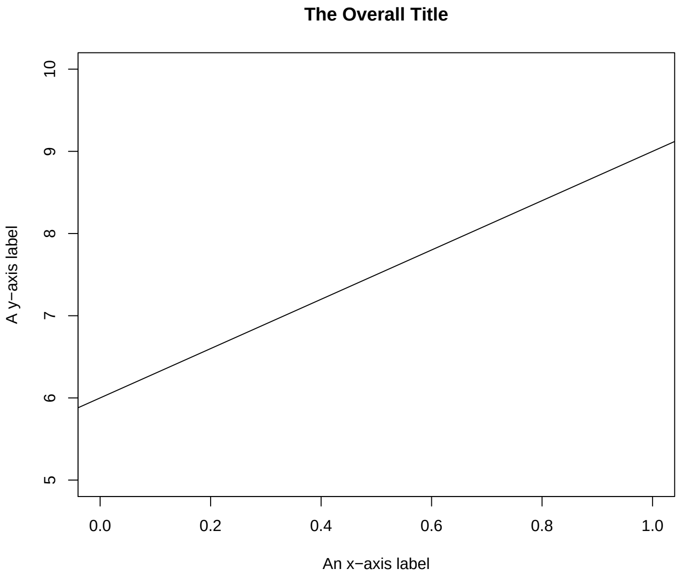
Statistics 120
Graphics II

Building Up Plots

- Graphs are produced in R by calling functions which build up graphs in a step-by-step fashion.
- Each function call carried out one small step of producing the final graph.


The Overall Title
A y−axis label
An x−axis label
0.0
0.2
0.4
0.6
0.8
1.0
5
6
7
8
9
10

Producing The Graph

Here is the code which produces the previous graph.

```
plot.new()
plot.window(xlim = c(0, 1), ylim = c(5, 10))
abline(a = 6, b = 3)
axis(1)
axis(2)
title(main = "The Overall Title")
title(xlab = "An x-axis label")
title(ylab = "A y-axis label")
box()
```

The Steps

- `plot.new()` signals to R that a new plot is to be produced.
- The `plot.window()` call sets the limits for the *x* and *y* coordinates in the graph.
- The `abline()` call draws a line with intercept 6 and slope 3 across the graph.
- `axis(1)` draws the *x*-axis.
- `axis(2)` draws the *y*-axis.
- Calls to `title()` are used to add annotation.
- `box()` draws a box around the graph.

Drawing Primitives

- Adding Points To A Plot
- Adding Connected Line Segments To A Plot
- Drawing Straight Lines Across A Plot
- Adding Disconnected Lines Segments To A Plot
- Adding Arrows To A Plot
- Adding Rectangles To A Plot
- Adding Polygons To A Plot
- Adding Text To A Plot

Adding Points To A Plot

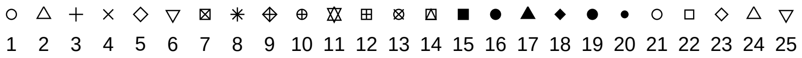
The basic call has the form:

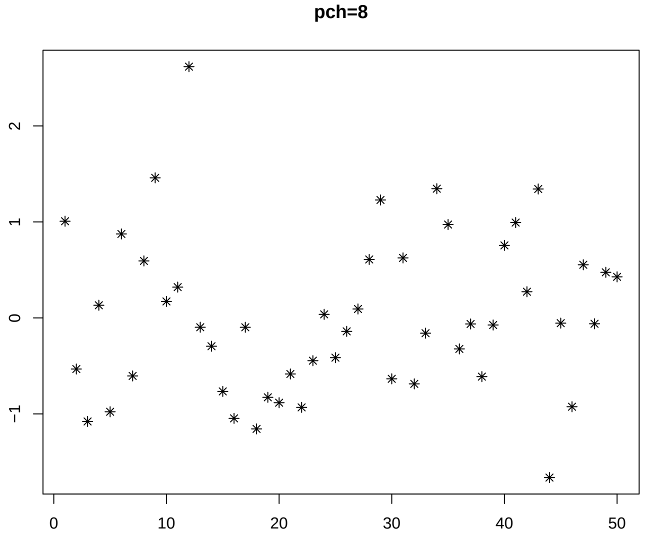
`points(x, y, pch=`***int***`, col=`***str***`)`

where

- `pch=`***int*** specifies the plotting symbol. Values 1 to 25 are special graphical symbols, values from 33 to 126 are taken to ASCII codes. A quoted character will also work.
- `col=`***str*** is a colour specification. Examples, `"red"`, `"lightblue"`, etc. (More on colour later).

Graphical Plotting Symbols




pch=8
2
1
0
−1
0
10
20
30
40
50

Adding Connected Line Segments To A Plot

The basic call has the form:

`lines(x, y, lty=`***str***`, lwd=`***num***`, col=`***str***`)`

where

- `lty=`***int*** specifies the line texture. One of `"blank"`, `"solid"`, `"dashed"`, `"dotted"`, `"dotdash"`, `"longdash"` or `"twodash"`.

  Alternatively the length of on/off penstokes in the texture. `"11"` is a high density dotted line, `"33"` is a short dashed line and `"1333"` is a dot-dashed line.

- `lwd=`***num*** and `col=`***str*** specify the line width and colour.

Drawing Straight Lines Across A Plot

The basic call has the forms:

`abline(a=`*intercept*`, b=`*slope*`)`
`abline(h=`*numbers*`)`
`abline(v=`*numbers*`)`

where

- The `a`/`b` form specifies a line in slope intercept form.
- `h=` specifies horizonal lines at the given *y* values.
- `v=` specifies vertical lines at the given *x* values.
- Line texture, colour and width arguments can also be given.

Adding Disconnected Lines Segments To A Plot

The basic call has the form:

```
segments(x0, y0, x1, y1)
```

where

- The `x0`, `y0`, `x1`, `y1` arguments give the start and end coordinates of the segments.
- Line texture, colour and width arguments can also be given.

Adding Arrows To A Plot

The basic call has the form:

```
arrows(x0, y0, x1, y1, code=int,
       length=num, angle=num)
```

where

- The `x0`, `y0`, `x1`, `y1` arguments give the start and end coordinates of the arrows.
- `code=1` – head at the start, `code=2` – head at the end and `code=3` – a head at both ends.
- `length` and `angle` – length of the arrow head and angle to the shaft.

Adding Rectangles To A Plot

The basic call has the form:

`rect(x0, y0, x1, y1, col=`*str*`, border=`*str*`)`

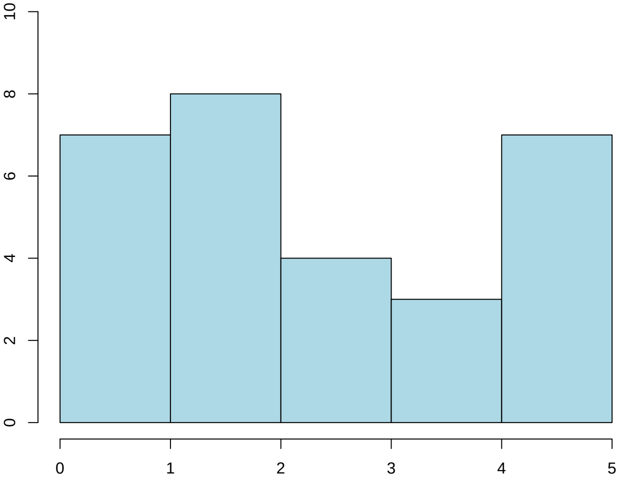
where

- `x0`, `y0`, `x1`, `y1` give the coordinates of diagonally opposite corners of the rectangles.
- `col` and `border` specify the colour of the interior and border of the rectangles.
- line texture and width specifications can also be given.

Rectangle Example

The following code illustrates how a barplot or histogram could be constructed.

```
plot.new()
plot.window(xlim = c(0, 5), ylim = c(0, 10))
rect(0:4, 0, 1:5, c(7, 8, 4, 3), col="lightblue")
axis(1)
axis(2)
```


10
8
6
4
2
0
0
1
2
3
4
5

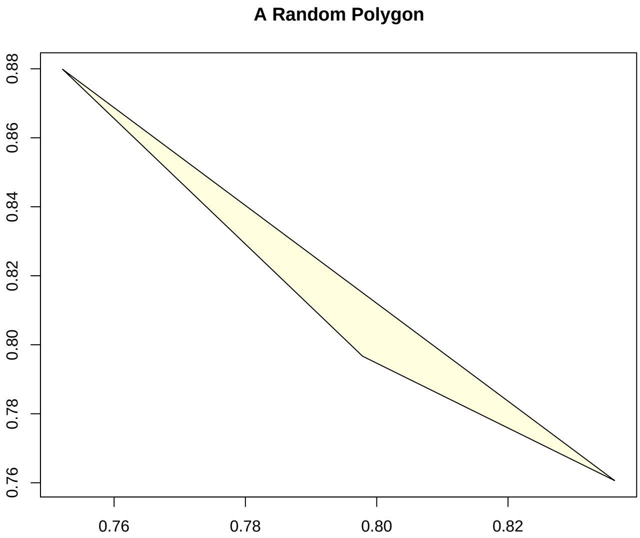
Adding Polygons To A Plot

The basic call has the form:

`polygon(x, y, col=`*str*`, border=`*str*`)`

where

- `x` and `y` give the coordinates of the polygon vetexes. `NA` values separate polygons.
- `col` and `border` specify the colour of the interior and border of the polygons.
- line texture and width specifications can also be given.


A Random Polygon
0.88
0.86
0.84
0.82
0.80
0.78
0.76
0.76
0.78
0.80
0.82

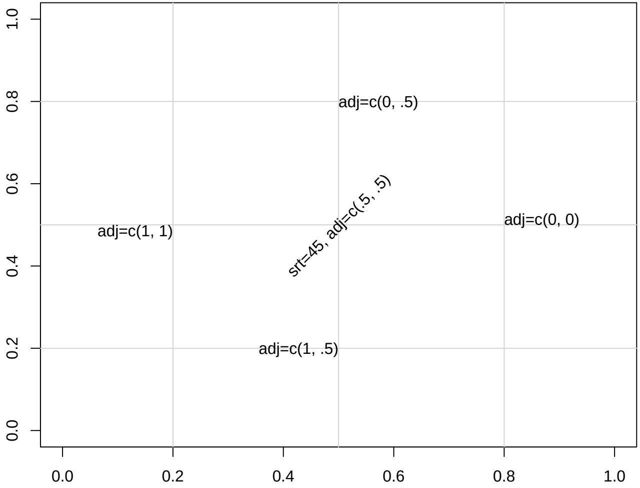
Adding Text To A Plot

The basic call has the form:

```
text(x, y, labels)
```

where

- `x` and `y` give the coordinates where the text is to be placed.
- `labels` gives the actual text strings.
- `col` gives the colour of the text.
- `srt` gives the rotation of the strings (counterclockwise in degrees from the horizontal).
- `adj` gives the justification of the strings.


adj=c(0, .5)
srt=45, adj=c(.5, .5)
adj=c(1, 1)
adj=c(0, 0)
adj=c(1, .5)
0.0
0.2
0.4
0.6
0.8
1.0
0.0
0.2
0.4
0.6
0.8
1.0

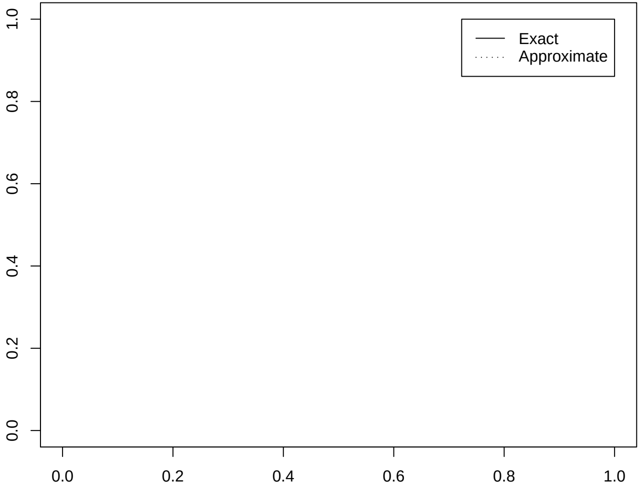
Adding A Legend To A Plot

A simple example has the form:

```
legend(xloc, yloc,
       legend = c("Exact", "Approximate"),
       lty = c("solid", "dotted"),
       xjust = 1, yjust = 1)
```

where

> `xloc` and `yloc` give the coordinates where the legend is to be placed. `legend` gives the text strings. `lty` gives line textures. `xjust` and `yjust` gives the justification of the legend box with respect to the location.



Exact
Approximate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

Axes and Annotation

The axis command can be customised.

```
axis(1, at=1:4, lab=c("A", "B", "C", "D"))
```

places the tick marks on the lower *x* axis at 1, 2, 3, and 4 and labels them with the strings “A”, “B”, “C” and “D”.

Label rotation can be controlled with `las=`. Setting `las=0` produces labels which are placed parallel to their axes, `las=1` produces labels which are horizontally oriented, `las=2` produces labels which are at right-angles to the axis and `las=3` produces labels which are vertically oriented.

Manipulating the Axis Limits

The statement

```
plot.window(xlim=c(0,1), ylim=c(10,20))
```

produces axis limits which are expanded by 6% over those actually specified. This expansion can be inhibited by specifying `xaxs="i"` and/or `yaxs="i"`.

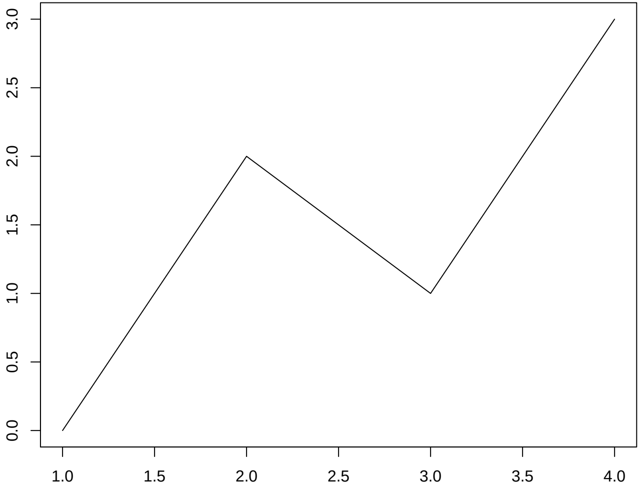
For example, the call

```
plot.window(xlim=c(0,1), ylim=c(10,20),
            xaxs="i")
```

produces a plot with 0 lying at the extreme left of the plot region and 1 lying at the extreme right.

Drawing Line Graphs

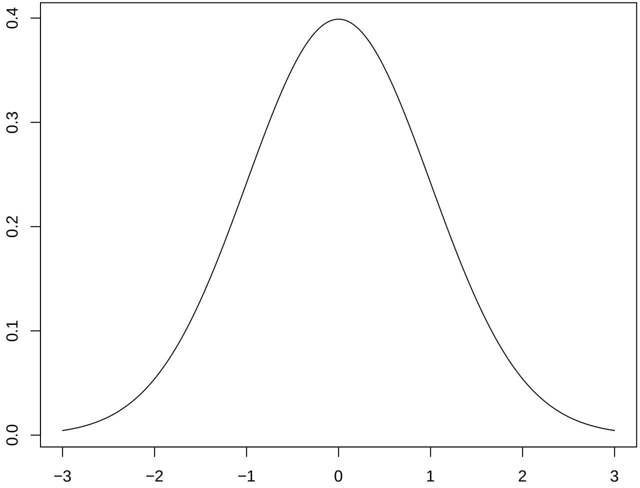
```
plot.new()
plot.window(xlim = c(1, 4), ylim = c(0, 3))
x = c(1, 2, 3, 4)
y = c(0, 2, 1, 3)
lines(x, y)
axis(1)
axis(2)
box()
```


0.0
0.5
1.0
1.5
2.0
2.5
3.0
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Drawing Curves

One of the most common graphics tasks is to draw the graph of $y = f(x)$ over an interval $[a, b]$. One way to do this is to approximate the graph by a series of straight line segments. For example, we could draw a graph of the density of the normal distribution as follows.

```
x = seq(-3, 3, length = 1000)
y = dnorm(x)
plot.new()
plot.window(xlim = range(x), ylim = range(y))
lines(x, y)
axis(1)
axis(2)
box()
```



0.0
0.1
0.2
0.3
0.4
−3
−2
−1
0
1
2
3

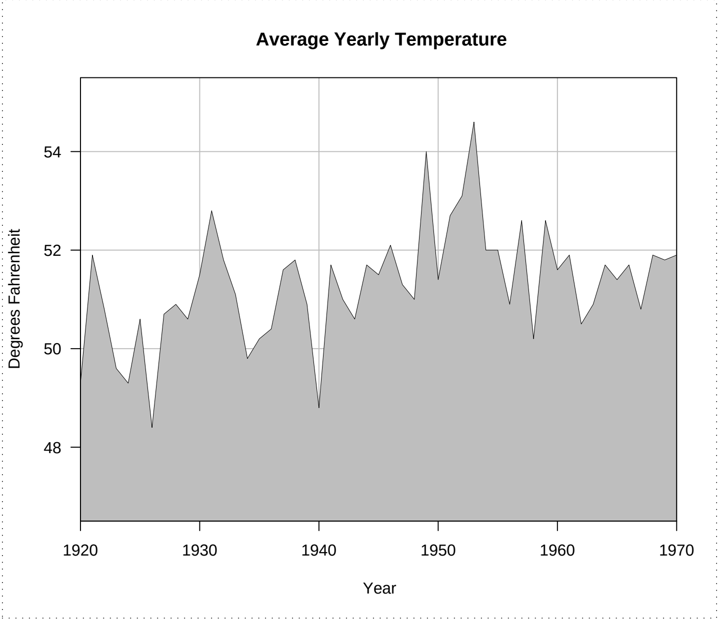
Filling Areas In Line Graphs

Annual year temperatures in New Haven (1920-1970).

```
> y
 [1] 49.3 51.9 50.8 49.6 49.3 50.6 48.4 50.7 50.9
[10] 50.6 51.5 52.8 51.8 51.1 49.8 50.2 50.4 51.6
[19] 51.8 50.9 48.8 51.7 51.0 50.6 51.7 51.5 52.1
[28] 51.3 51.0 54.0 51.4 52.7 53.1 54.6 52.0 52.0
[37] 50.9 52.6 50.2 52.6 51.6 51.9 50.5 50.9 51.7
[46] 51.4 51.7 50.8 51.9 51.8 51.9
```

The corresponding years.

```
> x = 1920:1970
```


Average Yearly Temperature
Degrees Fahrenheit
48
50
52
54
1920
1930
1940
1950
1960
1970
Year

Producing The Plot

Setting up the plot and drawing the background grid.

```
plot.new()
plot.window(xlim = c(1920, 1970), xaxs = "i",
            ylim = c(46.5, 55.5), yaxs = "i")

abline(v = seq(1930, 1960, by = 10), col = "gray")
abline(h = seq(48, 54, by = 2), col = "gray")
```

Drawing the filled polygon.

```
xx = c(1920, x, 1970)
yy = c(46.5, y, 46.5)
polygon(xx, yy, col = "gray")
```

Producing The Plot

Finishing off.

```
axis(1)
axis(2, las = 1)
box()
title(main = "Average Yearly Temperature")
title(ylab = "Degrees Fahrenheit")
title(xlab = "Year")
```