Statistics 120
Graphics

Computer Graphics

- Drawing graphics in a window on the screen of a computer is very similar to drawing by hand on a sheet of paper.
- We begin a drawing by getting out a clean piece of paper and then deciding what scale to use in the drawing.
- With those basic decisions made, we can then start putting pen to paper.
- The steps in R are very similar.

Starting a New Plot

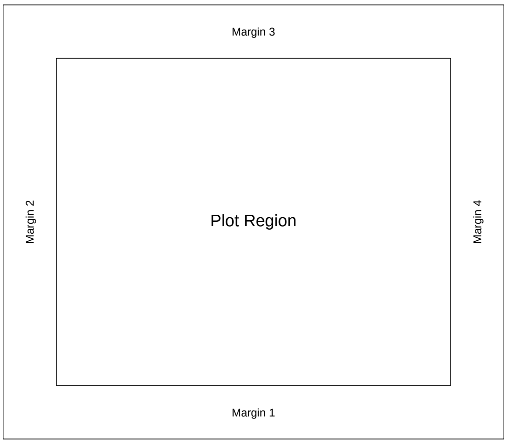
We begin a plot by first telling the graphics system that we are about to start a new plot.

```
> plot.new()
```

This indicates that we are about to start a new plot and must happen before any graphics takes place.

The call to `plot.new` chooses a default rectangular region for the plot to appear in. This choice can be overridden using the `par` function.

The plotting region is surrounded by four *margins*.



Controlling The Margins

There are a variety of ways of setting the sizes of the plot margins using `par`.

1. Set the margin sizes in inches.

   ```
   > par(mai = c(2, 2, 1, 1))
   ```

2. Set the margin sizes in lines of text.

   ```
   > par(mar = c(4, 4, 2, 2))
   ```

3. Set the plot width and height in inches.

   ```
   > par(pin = c(5, 4))
   ```

Setting the Axis Scales

Next we set the scales on along the sides of the plot. This determines how coordinates get mapped onto the page.

```
plot.window(xlim = xlimits, ylim = ylimits)
```

The graphics system arranges for the specified region to appear on the page.

`xlimits` and `xlimits` are vectors which contain the lower and upper limits to appear on the *x* and *y* axes.

For example,

```
   xlim = c(-pi, pi), ylim = c(-1, 1),
```

might be suitable for plotting sine and cosine functions.

Setting the Axis Scales

There is also an optional argument to the function `plot.window()` which allows a user to specify a particular aspect ratio.

```
> plot.window(xlim = xlimits, ylim = ylimits,
      asp = 1)
```

The use of `asp=1` means that unit steps in the *x* and *y* directions produce equal distances in the *x* and *y* directions on the page.

This is important if circles are to appear as circles rather than ellipses.

Drawing

With the plot setup done, we can now draw on the page. There are a number of R functions which can be used to draw. The simplest of these are:

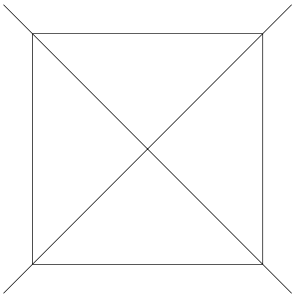
`points`	draw “points” on a plot
`lines`	draw connected line segments
`segments`	draw disconnected line segments
`rect`	draw rectangles
`polygon`	draw filled polygons
`text`	draw text on a plot
`box`	draw a box around a plot

Square with Diagonals Example

These commands draw a square with a cross drawn across its diagonals.

```
> plot.new()
> plot.window(xlim = c(0, 1),
              ylim = c(0, 1), asp  =  1)
> rect(xleft = .1, ybottom = .1,
       xright = .9, ytop = .9)
> segments(0, 0, 1, 1)
> segments(0, 1, 1, 0)
```

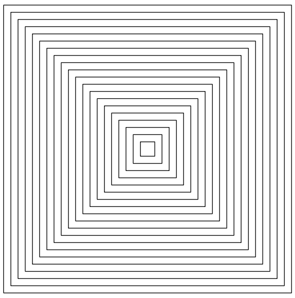
A Square with Diagonals



Nested Squares Example

This example shows how to draw a set of nested squares. Note that all 21 squares are produced by a single call to `rect`.

```
> plot.new()
> plot.window(xlim = c(0, 1), ylim = c(0,
      1), asp = 1)
> p = seq(0, 0.5, length = 21)
> rect(p, p, 1 - p, 1 - p)
```

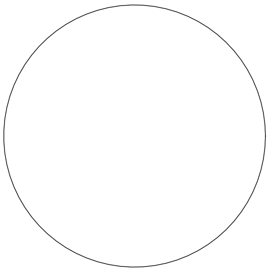


Drawing a Circle

There is no simple R function for drawing a circle. Here is how it can be done by approximating the circle with a regular polygon.

```
> plot.new()
> plot.window(xlim = c(-1.1, 1.1), ylim = c(-1.1,
      1.1), asp = 1)
> theta = seq(0, 2 * pi, length = 72)
> x = cos(theta)
> y = sin(theta)
> lines(x, y)
```

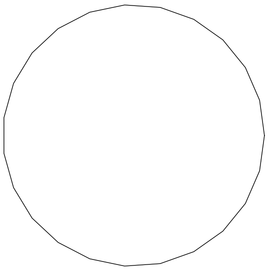
Change of angle = 5°



Approximating Smooth Curves

- Suppose that a series of connected line segments is to be used to approximate a smooth curve.
- Provided that the lines change direction by no more than 5°, then they will appear to the eye to make up a smooth curve.
- This is why 72 line segments were used in the previous example — 360 equals 72 times 5!

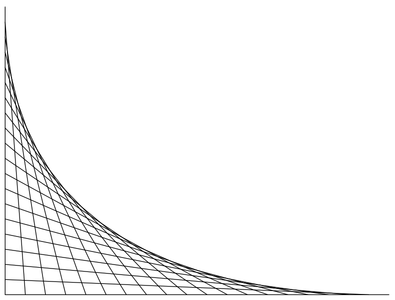
Change of angle = 15°



Another Curve Example

Here is another example which shows how the eye can perceive a sequence of straight lines as a curve.

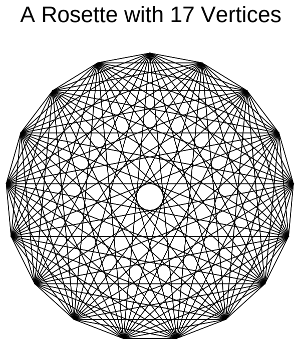
```
> x1 = seq(0, 1, length = 20)
> y1 = rep(0, 20)
> x2 = rep(0, 20)
> y2 = seq(0.75, 0, length = 20)
> plot.new()
> plot.window(xlim = c(0, 1), ylim = c(0,
      0.75), asp = 1)
> segments(x1, y1, x2, y2)
```



Rosettes

A rosette is a figure which is created by taking a series of equally spaced points around the circumference of a circle and joining each of these points to all the other points.

```
> n = 17
> theta = seq(0, 2 * pi, length = n + 1)[1:n]
> x = sin(theta)
> y = cos(theta)
> v1 = rep(1:n, n)
> v2 = rep(1:n, rep(n, n))
> plot.new()
> plot.window(xlim = c(-1, 1), ylim = c(-1,
      1), asp = 1)
> segments(x[v1], y[v1], x[v2], y[v2])
```



Drawing a Spiral

- A spiral is created by drawing around the outside of a circle whose radius is increasing:

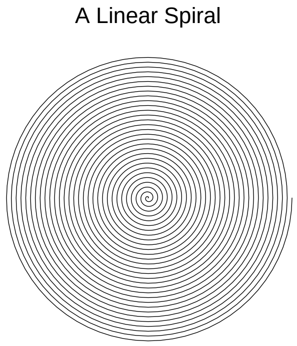
$$\begin{aligned} x_t &= R_t \cos\theta t \\ y_t &= R_t \sin\theta t \end{aligned}$$

- The radius is an increasing function of t.

Drawing a Spiral

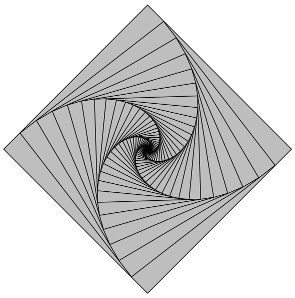
These commands draw a spiral, centred on (0,0). The spiral does 30 revolutions:

```
> theta = seq(0, 30 * 2 * pi, by = 2 * pi/72)
> x = cos(theta)
> y = sin(theta)
> R = theta/max(theta)
> plot.new()
> plot.window(xlim = c(-1, 1), ylim = c(-1,
      1), asp = 1)
> lines(x * R, y * R)
```



Spiral Squares

```
> plot.new()
> plot.window(xlim = c(-1, 1), ylim = c(-1,
      1), asp = 1)
> square = seq(0, 2 * pi, length = 5)[1:4]
> n = 51
> r = rep(1.12, n)
> r = cumprod(r)
> r = r/r[n]
> theta = seq(0, 2 * pi, length = n)
> for (i in n:1) {
      x = r[i] * cos(theta[i] + square)
      y = r[i] * sin(theta[i] + square)
      polygon(x, y, col = "gray")
  }
```



Drawing a Scatter Plot

- With the tools we have at hand, we are now in a position to built a new tool for producing scatter plots.
- There are a number of tasks which must be solved:
  - Determining the *x* and *y* ranges.
  - Setting up the plot window.
  - Plotting the points.
  - Adding the plot axes and frame.

Scatter Plot Code

Here are the steps required to produce a scatter plot.

- Determine the *x* and *y* ranges.

  ```
  > xlim = range(x)
  > ylim = range(y)
  ```

- Set up the plot window.

  ```
  > plot.new()
  > plot.window(xlim = xlim, ylim = ylim)
  ```

- Plot the points.

  ```
  > points(x, y)
  ```

A Scatter Plot Function

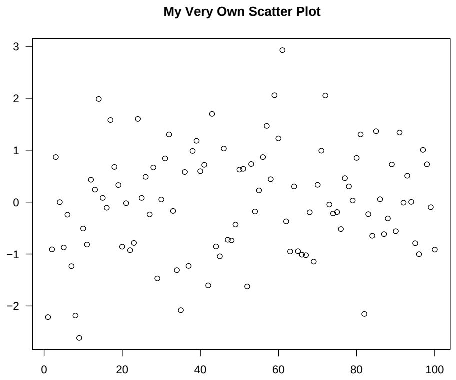
By “wrapping” the steps in a function definition we can produce a simple scatter plot function.

```
> scat = function(x, y) {
      xlim = range(x)
      ylim = range(y)
      plot.new()
      plot.window(xlim = xlim, ylim = ylim)
      points(x, y)
      axis(1)
      axis(2)
      box()
  }
```

A Scatter Plot Function

We can use this function just like any other R function to produce scatter plots.

```
> xv = 1:100
> yv = rnorm(100)
> scat(xv, yv)
> title(main = "My Very Own Scatter Plot")
```


My Very Own Scatter Plot