CSC 121: Computer Science for Statistics

Radford M. Neal, University of Toronto, 2017

http://www.cs.utoronto.ca/∼radford/csc121/

Week 9

Operations on Numeric Vectors that Produce One Number

R has several functions that take a numeric vector or matrix as their argument, and return a single number as their value, including:

`sum`	finds the sum of all elements.
`prod`	finds the product of all elements.
`max`	finds the largest of all elements.
`min`	finds the smallest of all elements.
`mean`	finds the mean (average) of all elements.

For example:

```
> u <- c(3,5,1,9)
> sum(u)
[1] 18
```

This does pretty much the same thing as the following loop:

```
> s <- 0
> for (x in u) s <- s + x
> s
[1] 18
```

However, `sum(u)` is faster, and in some cases more accurate.

Operations on Logical Vectors that Produce One Logical Value

R also has two functions that take a logical vector as their argument, and return a single logical value:

`any`	Return `TRUE` if any elements are `TRUE`
`all`	Return `TRUE` if all elements are `TRUE`

Looked at another way, `any` finds the "or" of all elements in its argument, and `all` finds the "and" of all elements.

Here's an example of the use of these functions:

```
check_age <- function (df) {
    if (any(is.na(df$age)))
        stop("Age is missing for some people")
    if (!all (df$age >= 0 & df$age < 150))
        stop("Age is invalid for some people")
}
```

Can you think of a way to replace the second `if` condition with one that uses `any` rather than `all`?

Creating a Plot in Stages

Many simple plots can be created with a single `plot` command — eg, `plot(x,y)` will plot points with coordinates given by the vectors `x` and `y`.

More complicated plots can be created in stages by adding more points, lines, and text to what has already been plotted.

The general approach:

- Create a new plot with `plot`. It might contains some points or lines, or might be completely empty. Features such as the axis scales and labels are determined at this stage.

- Then add more information, using functions such as `points`, `lines`, `abline`, and `text`. You can call these functions as many times as needed, perhaps with different options for things like colour and line width each time.

- You can also add a title above the plot with the `title` function.

Creating a New Plot

You create a new plot with the `plot` function. It takes one or two data vectors as its first arguments, but has many, many other possible arguments. You'll want to let most of these have their default values, and refer to any that you set by name.

Here are some of the possible arguments to `plot`:

`type`	Type of plotting — `"p"` for points (the default), `"l"` for lines, `"b"` for both points and lines, `"c"` for lines only but with space for points
`col`	Colour for points/lines plotted (default is `"black"`)
`xaxt`	Set to `"n"` to get rid of horizontal axis numbers
`yaxt`	Set to `"n"` to get rid of vertical axis numbers
`xlab`	Label for the horizontal axis
`ylab`	Label for the vertical axis
`xlim`	Horizontal range for plot (vector of length two)
`ylim`	Vertical range for plot (vector of length two)
`asp`	Aspect ratio, `asp=1` ensures one vertical unit looks the same length as one horizontal unit

For example, `plot (c(), xlim=c(0,2), ylim=c(1,5))` will plot an empty frame with horizonal axis labels from 0 to 2 and vertical axis labels from 1 to 5.

Adding Points to a Plot

We can add points to a plot with the `points` function. Like `plot`, it takes two vectors as its first two arguments, containing the x and y coordinates of the points. (Or just a single vector argument with the y coordinates, in which case the x coordinates are 1, 2, 3, ...)

It can also take other arguments that set various options, such as

`type`	Set to `"b"` for lines as well as points
`col`	Colour for points plotted
`pch`	Character to plot points with — default is a circle, other possibilities are `pch="x"` for plotting with x symbols, or `pch=20` for solid dots

For example, `points (x, y, col="red", pch=20)` will add solid red dots to the plot, at the coordinates given by the vectors `x` and `y`.

Adding Lines to a Plot

We can add lines to a plot with the `lines` function.

In addition to one or two arguments giving the coordinates of the points to connect with lines, it can take other arguments such as those below (which can also be used for `plot`):

`type`	Set to `"b"` for points too, `"c"` for lines only but with space for points
`col`	Colour for lines plotted
`lty`	Line type — eg, `"dotted"`, `"dashed"`, or `"solid"` (the default)
`lwd`	Line width (default is 1)

For example, `lines (y, col="green", lty="dotted")` will add dotted green lines to the plot, at the x coordinates 1, 2, 3, ... and y coordinates given by the vector `y`.

Adding Text to a Plot

We can add text to a plot with the `text` function.

Here's an example that adds "WOW" to the origin of the plot:

```
> text (0, 0, "WOW")
```

We can put many character strings on a plot with one call of `text`, since its arguments can be vectors of x coordinates, y coordinates, and character strings.

For example:

```
> x <- 1:10
> y <- x^2
> plot(x,y,xlim=c(0,11))
> text(x,y+2,paste("square of",x))
```

Example: Drawing a Spiral

Here's an example R script that draws a spiral in a plain box, using 7 segments each time it winds around, with red dots at the vertices. The start and end are labelled with "start" and "end".

```
n <- 20
angle <- 2*pi*(0:n)/7
dist <- 0:n
x <- dist * cos(angle)
y <- dist * sin(angle)

plot (x, y, type="c", xaxt="n", yaxt="n", xlab="", ylab="",
            xlim=c(-n,n), ylim=c(-n,n), asp=1)

points (x, y, col="red")

text (x[1], y[1]-1, "start")
text (x[n+1], y[n+1]+1, "end")
```

The Spiral Plot

```
> source("http://www.cs.utoronto.ca/~radford/csc121/spiral-script.r")
```