S3 Class/Methods for Polygons

Biostatistics 140.776

Defining the class

The S3 system doesn't have a formal way to define a class—there's no version of the `setClass()` function. But typically, we use a list to define the class, and elements of the list serve as slots or data elements.

Here is our definition of a polygon represented using Cartesian coordinates. The class contains an element called `xcoord` and `ycoord` for the x- and y-coordinates, respectively. The `make_poly()` function is the "constructor" function for polygon objects. It takes as arguments a numeric vector of x-coordinates and a corresponding numeric vector of y-coordinates.

```r
## Constructor function for polygon objects
## x a numeric vector of x coordinates
## y a numeric vector of y coordinates

make_poly <- function(x, y) {
        if(length(x) != length(y))
                stop("'x' and 'y' should be the same length")

        ## Create the "polygon" object
        object <- list(xcoord = x, ycoord = y)

        ## Set the class name
        class(object) <- "polygon"
        object
}
```

Defining the methods

Now that we have a class definition, we can develop some methods for operating on objects from that class.

The first method we'll define is the `plot` method, which will simply plot the polygon on the graphics device. It will do this by

1. Plotting the vertices on to "blank" plot for the purpose of setting up the coordinate system as well as the labels, etc.
2. Connect the vertices with lines to draw the polygon

Because a polygon must be closed (i.e. make a complete loop) we have to add a "fake" vertex on the the end, which is simply the first vertex repeated. That way when we call the `lines()` function, it will be a completely connected object.

```r
## Plot method for polygon objects
## x an object of class "polygon"
## ... other arguments to be passed to the default plot method

plot.polygon <- function(x, y, ...) {
        ## Make a blank plot (type = "n") to set the
        ## coordinate systemand the x- and y-coordinate ranges
        plot(x$xcoord, x$ycoord, type = "n", ...)

        ## Make the x- and y- coordinates "wrap around"
        xp <- c(x$xcoord, x$xcoord[1])
        yp <- c(x$ycoord, x$ycoord[1])

        ## Connect polygon vertices with lines
        lines(xp, yp)
}
```

The `print()` method should just show some simple information about the object and should not be too verbose. Just enough information that the user knows what the object is.

Here the `print()` method just shows the user how many vertices the polygon has. It is a convention for `print()` methods to return the object `x` invisibly.

```
## Print method for polygon objects
## x an object of class "polygon"

print.polygon <- function(x, ...) {
        cat("a polygon with", length(x$xcoord),
            "vertices\n")
        invisible(x)  ## Convention to return object invisibly
}
```

The summary method typically shows a bit more information and may even do some calculations. This summary method computes the ranges of the x- and y-coordinates.

The typical approach for `summary()` methods is to allow the summary method to compute something, but to *not* print something. The strategy is

1. The `summary()` method returns an object of class "summary."
2. There is a `print()` method for "summary." objects.

See below for how this is done.

```
## Summary method for polygon objects
## object an object of class "polygon"

summary.polygon <- function(object, ...) {
        object <- list(rng.x = range(object$xcoord),
                       rng.y = range(object$ycoord))
        class(object) <- "summary.polygon"
        object
}

## Print method for summary.polygon objects
## x an object of class "summary.polygon"

print.summary.polygon <- function(x, ...) {
        cat("x:", x$rng.x[1], "-->", x$rng.x[2], "\n")
        cat("y:", x$rng.y[1], "-->", x$rng.y[2], "\n")
        invisible(x)
}
```

Using the class

Now we can make use of our class and methods.

```
## Construct a new "polygon" object
x <- make_poly(1:4, c(1, 5, 2, 1))
```

We can use the `print()` and `summary()` methods

```
print(x)
```

```
a polygon with 4 vertices
```

```
out <- summary(x)
class(out)
```

```
[1] "summary.polygon"
```

```
print(out)
```

```
x: 1 --> 4
y: 1 --> 5
```

Because of auto-printing we can just call the `summary()` method and let the results auto-print.

```
summary(x)
```

```
x: 1 --> 4
y: 1 --> 5
```

Now, the plot method.

```
plot(x, xlab = "x", ylab = "y", main = "Here's a polygon!")
```

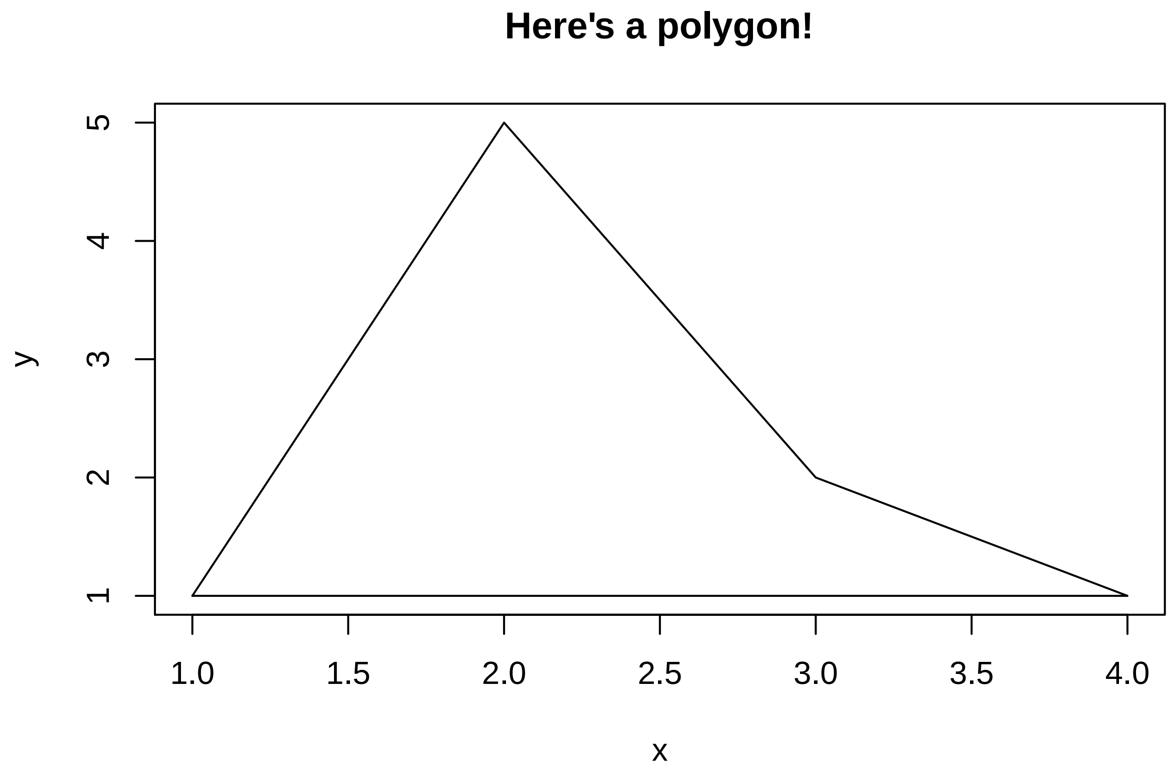

Figure 1: Plot of polygon object