Module 4: Lines and Polygons

CSCI E-5a: Programming in R

Let's clear the environment:

```r
rm( list = ls() )
```

Module Preview

Hello! And welcome to Module 4: Lines and Polygons.

In this module, we'll learn how to draw line segments and polygons.

- In Section 1, we'll see how to draw a line segment by using the `segments()` function.
- In Section 2, we'll learn how to modify the line type, width, and color for a line segment.
- In Section 3, we'll see how to use the `plot()` function to draw line segments.
- In Section 4, we'll learn how to draw a sequence of connected line segments by using the `lines()` function.
- In Section 5, we'll learn how to draw polygons by using the `polygon()` function.

When you've completed this module, you'll be able to:

- Draw a line segment on a pre-existing plotting region by using the `segments()` function.
- Adjust the line type, line width, and color of a line segment.
- Draw line segments by using the `plot()` function.
- Draw a connected sequence of line segments by using the `lines()` function.
- Create a polygonal shape by using the `polygon()` function.

In this module, we'll meet three new built-in R functions:

- `segments()`
- `lines()`
- `polygon()`

All right! Let's get started by learning how to draw line segments by using the `segments()` function.

Section 1: Line segments

Main Idea: *We can draw a line segment by using the `segments()` function*

In this section, we'll see how to draw a line segment by using the `segments()` function.

The `segments()` function draws a line segment from a starting point to an ending point.

This function takes a large number of input arguments, but four of them are absolutely necessary:

- The first input argument, denoted `x0`, is the x-coordinate of the starting point.
- The second input argument, denoted `y0`, is the y-coordinate of the starting point.
- The third input argument, denoted `x1`, is the x-coordinate of the ending point.
- The fourth input argument, denoted `y1`, is the y-coordinate of the ending point.

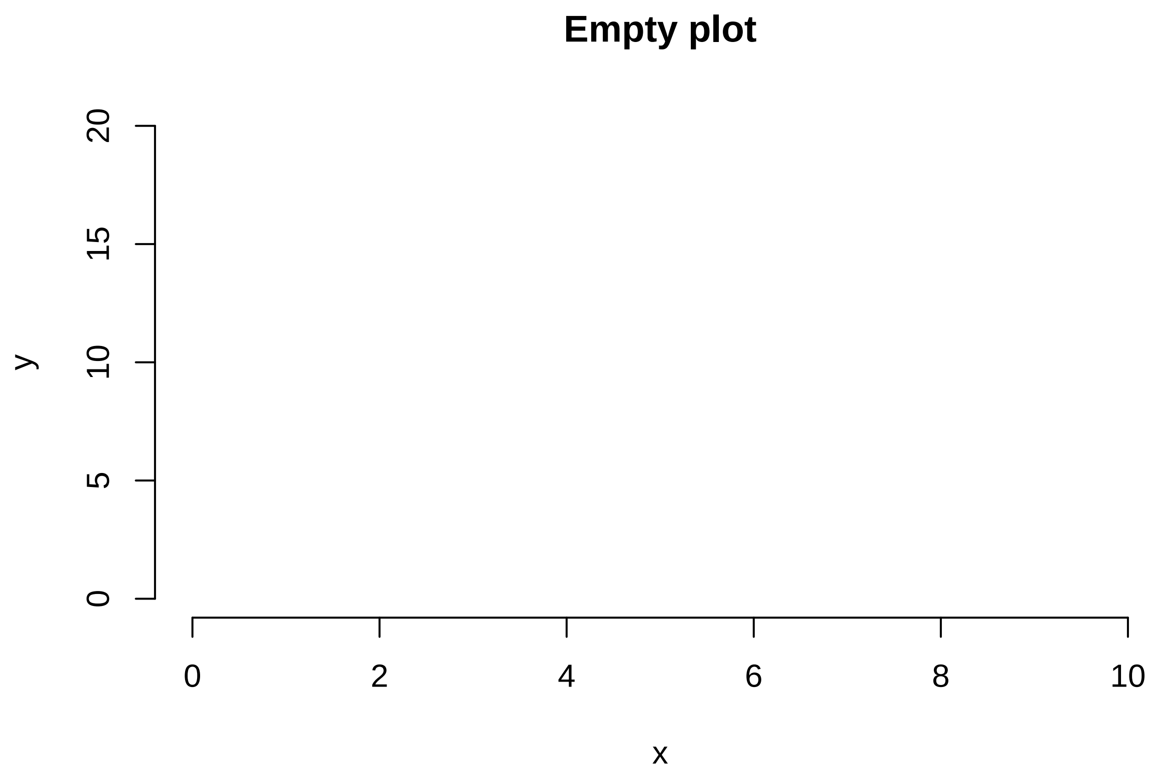
There's a catch to this: we can only use the `segments()` function with a pre-existing plotting region.

Thus, to draw a line segment, we have to first create an empty plot with no data and only then use the `segments()` function.

Let's see an example of this.

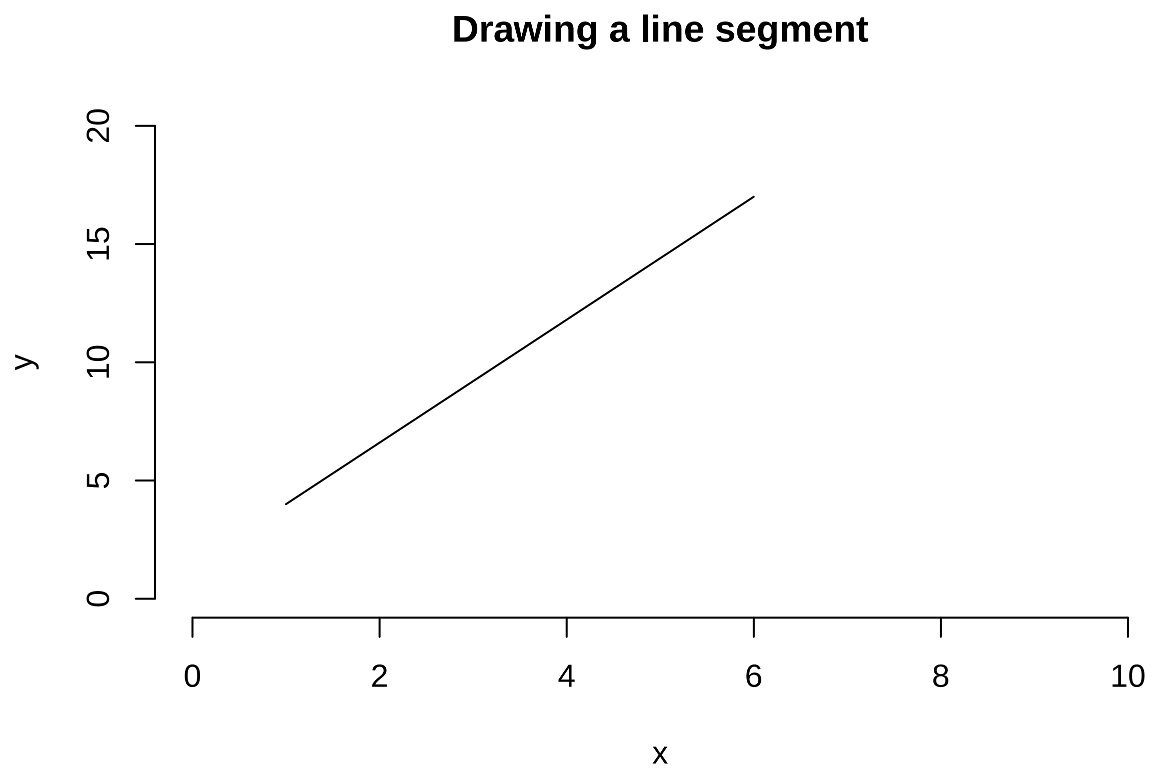
We'll first create an empty plot with no data, with the x-axis ranging from 0 to 10 and the y-axis ranging from 0 to 20:

```r
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Empty plot",
    xlab = "x",
    ylab = "y",
    bty = "n"
)
```



Now we'll draw a line segment starting at the point (1, 4) and ending at the point (6, 17), and we'll do this by creating the empty plot with no data first and then calling the `segments()` function:

```
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Drawing a line segment",
    xlab = "x",
    ylab = "y",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 1,
    y0 = 4,
    x1 = 6,
    y1 = 17
)
```



Amazing! Our first line segment graph!

So that's how to draw line segments by using the `segments()` function.

Now let's see how to modify a line segment.

Exercise 4.1: Draw a line segment

Create an empty plot with no data, with x ranging from 0 to 20 and y ranging from 0 to 10.

Then draw a line segment starting at the point (3, 7) and ending at the point (12, 4).

Solution

```
# Type your answer in here
```

Section 2: Modifying the line segment

Main Idea: *We can modify aspects of a line segment.*

In this section, we'll learn how to modify the line type, width, and color for a line segment.

Just as with points, we have many ways to modify the appearance of a line segment.

Changing the line width

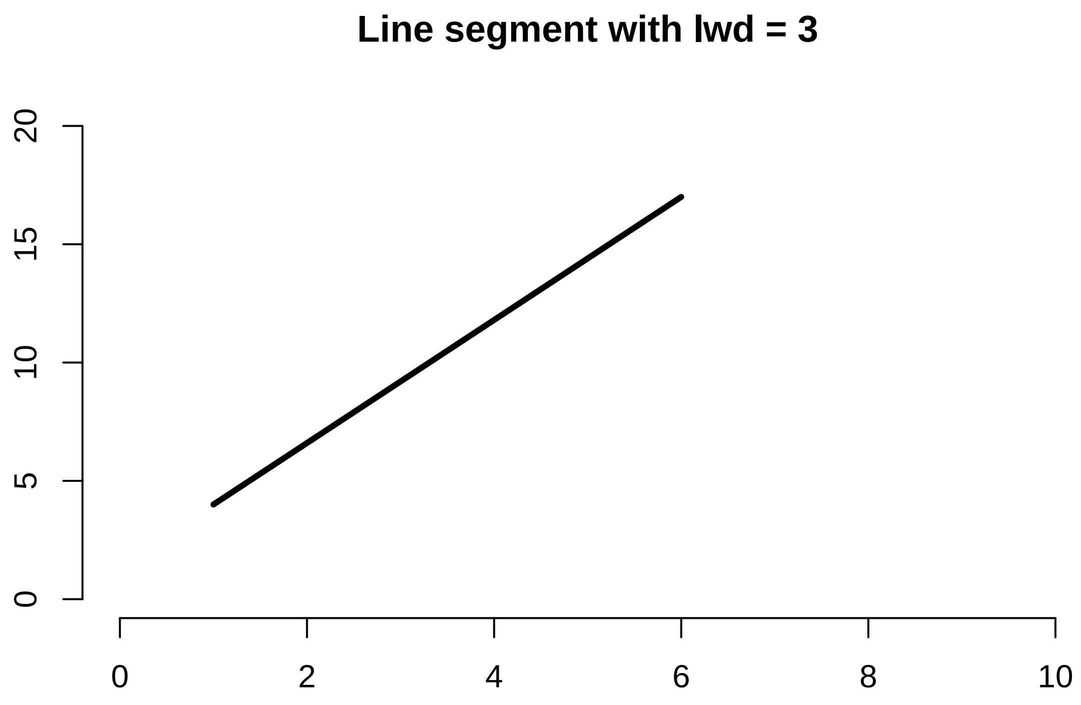
We can specify the width of the line by using the `lwd` input argument.

I find that line widths of 2 or 3 work nicely.

Let's first create an empty plot with no data, with the x-axis ranging from 0 to 10 and the y-axis ranging from 0 to 20.

Then we'll draw a line segment starting at (1, 4) and ending at (6, 17), with a line width of 3.

```r
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Line segment with lwd = 3",
    xlab = "",
    ylab = "",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 1,
    y0 = 4,
    x1 = 6,
    y1 = 17,
    lwd = 3   # We specify the line width here
)
```



Notice that the line is now a little thicker.

Changing the line type

The `segments()` function has an optional input argument named `lty`, and this controls the type of line that is drawn:

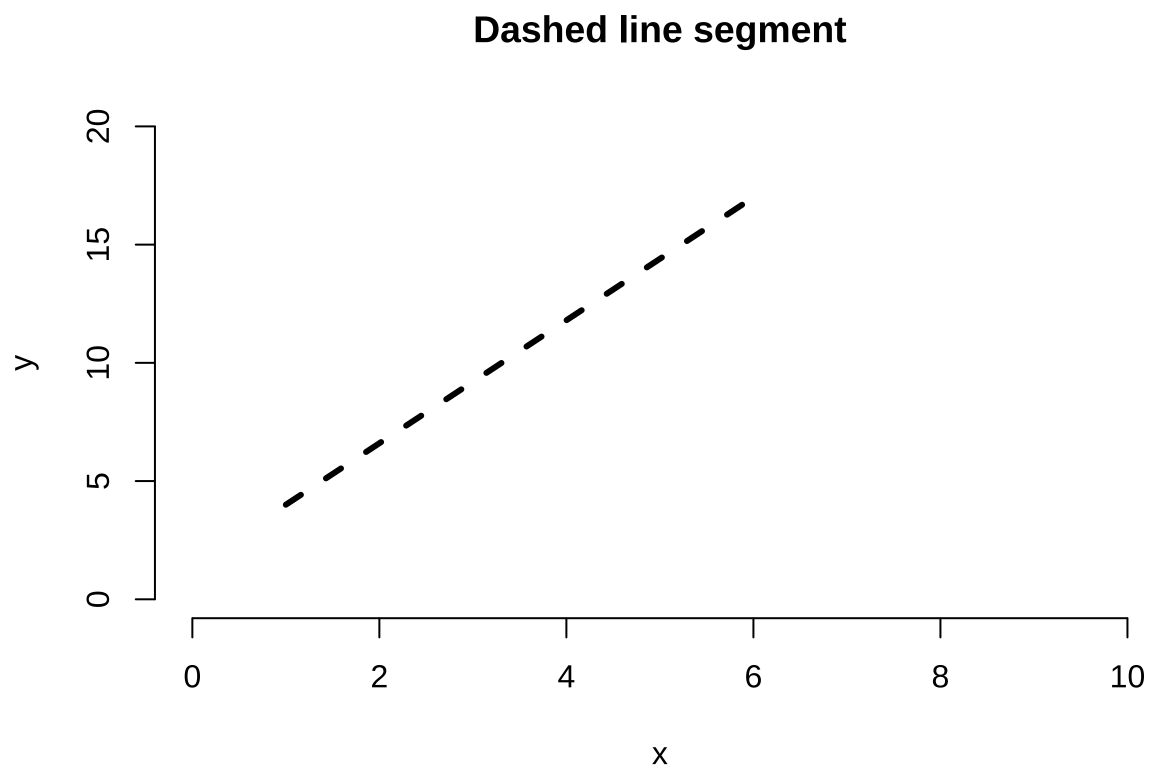
- If `lty = "solid"`, then a solid line is drawn.
- If `lty = "dashed"`, then a dashed line is drawn.
- If `lty = "dotted"`, then a dotted line is drawn.

The spacing between the dashes or dots depends on the line width, and in general I find it nice to use a line width of 2 or 3 for these lines.

Let's first create an empty plot with no data, with the x-axis ranging from 0 to 10 and the y-axis ranging from 0 to 20.

Now we'll draw a dashed line segment starting at (1, 4) and ending at (6, 17), with a line width of 3.

```
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Dashed line segment",
    xlab = "x",
    ylab = "y",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 1,
    y0 = 4,
    x1 = 6,
    y1 = 17,
    lwd = 3,
    lty = "dashed"  # Here we specify the line type is
                    # "dashed"
)
```

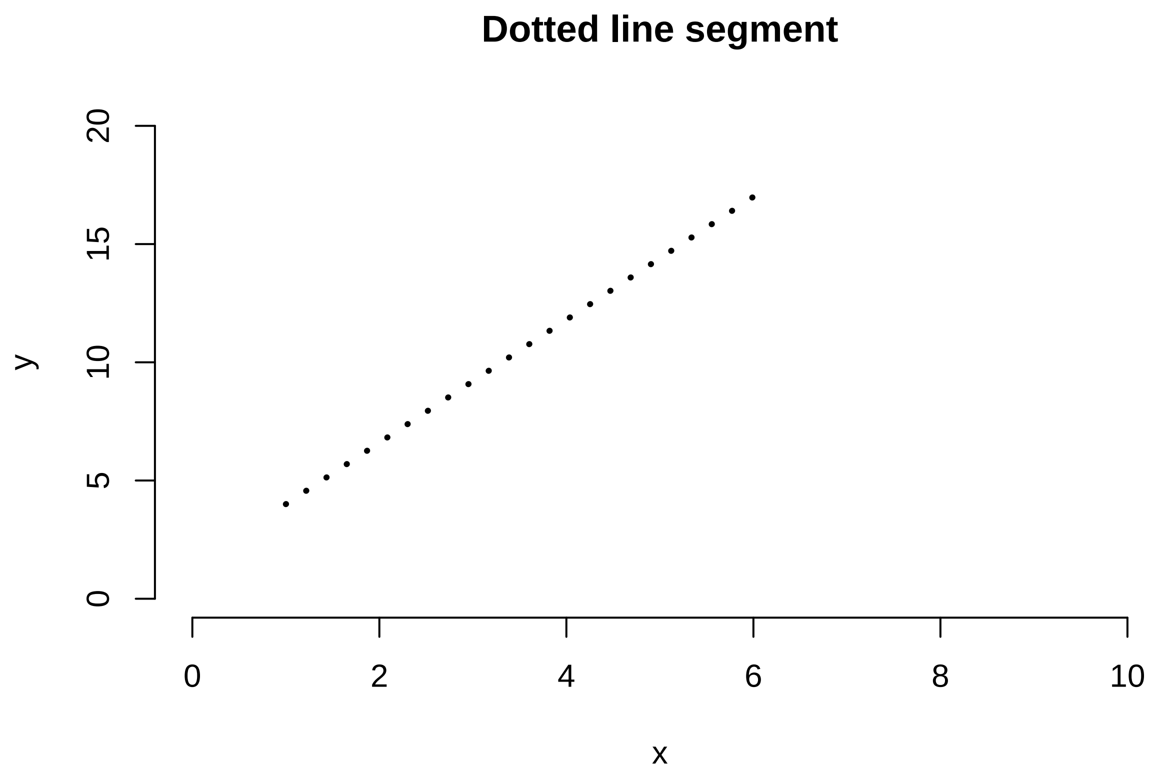


You can also specify the line type by using a numeric value:

Line Type	Numeric Value
Solid	1
Dashed	2

Line Type	Numeric Value
Dotted	3

```r
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Dotted line segment",
    xlab = "x",
    ylab = "y",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 1,
    y0 = 4,
    x1 = 6,
    y1 = 17,
    lwd = 3,
    lty = 3  # Here we use a numeric value to
             # specify that the line type is "dotted"
)
```



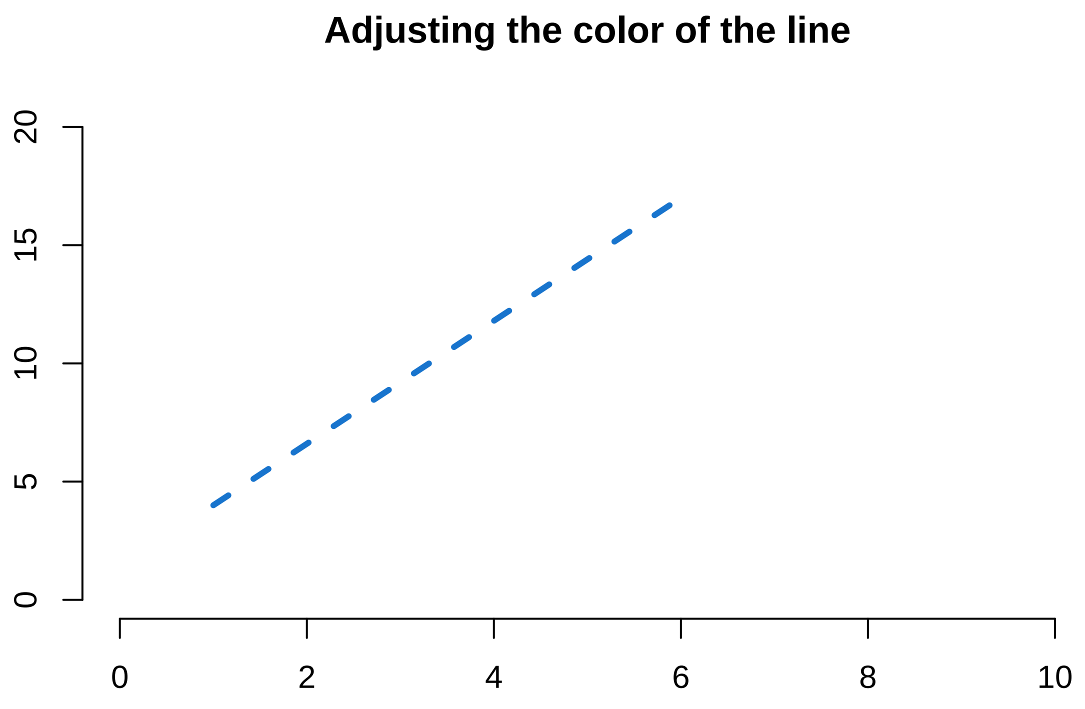
Adjusting the color of the line

We adjust the color of the line segment in the same way we do for points, by using the `col` optional input argument.

Let's first create an empty plot with no data, with the x-axis ranging from 0 to 10 and the y-axis ranging from 0 to 20.

Now we'll draw a dashed line segment starting at (1, 4) and ending at (6, 17), with a line width of 3 and a color of "dodgerblue3":

```r
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Adjusting the color of the line",
    xlab = "",
    ylab = "",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 1,
    y0 = 4,
    x1 = 6,
    y1 = 17,
    lwd = 3,
    lty = "dashed",
    col = "dodgerblue3"   # Here we specify the
                          # color of the line.
)
```



So that's how to how to modify the line type, width, and color for a line segment.

Now let's see how to use the `plot()` function to draw line segments.

Exercise 4.2: Line type, width, and color

Create an empty plot with no data, with x ranging from 0 to 20 and y ranging from 0 to 10.

Then draw a dotted line segment starting at the point (3, 7) and ending at the point (12, 4), using a line width of 3 and a color of "cadetblue3".

Solution

```r
# Type your answer here
```

Section 3: The `plot()` Function

Main Idea: *We can draw line segments by using the `plot()` function.*

In this section, we'll see how to use the `plot()` function to draw line segments.

Let's consider the line segment that starts at the point $x = 1$ and $y = 4$, and ends at the point $x = 6$ and $y = 17$.

We can write this in a table format:

	x	y
Start	1	4
Finish	6	17

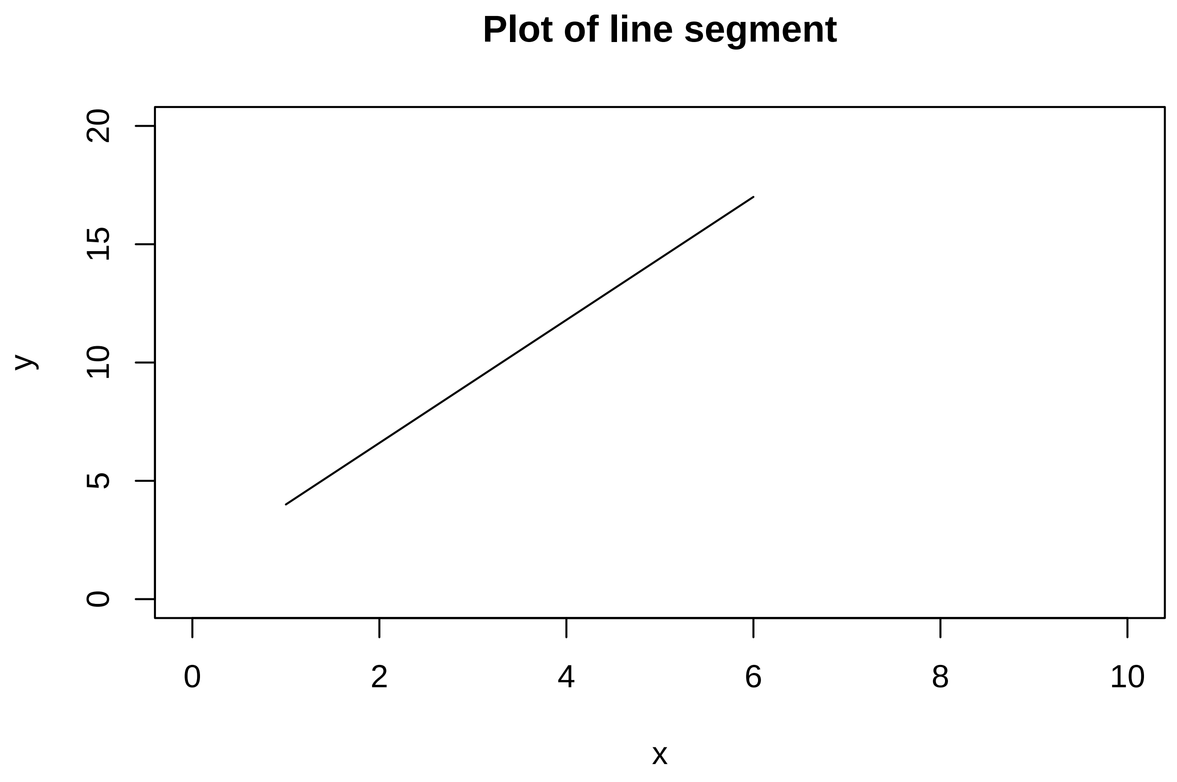
We saw with the `points()` function that we could plot multiple points by using a vector for all the x-coordinates and another vector for all the y-coordinates.

We can employ this same strategy with the `plot()` function to draw a line segment by using the optional input argument `type`.

If we set `type = "l"`, then the `plot()` function will create the graphical plot and then draw the line.

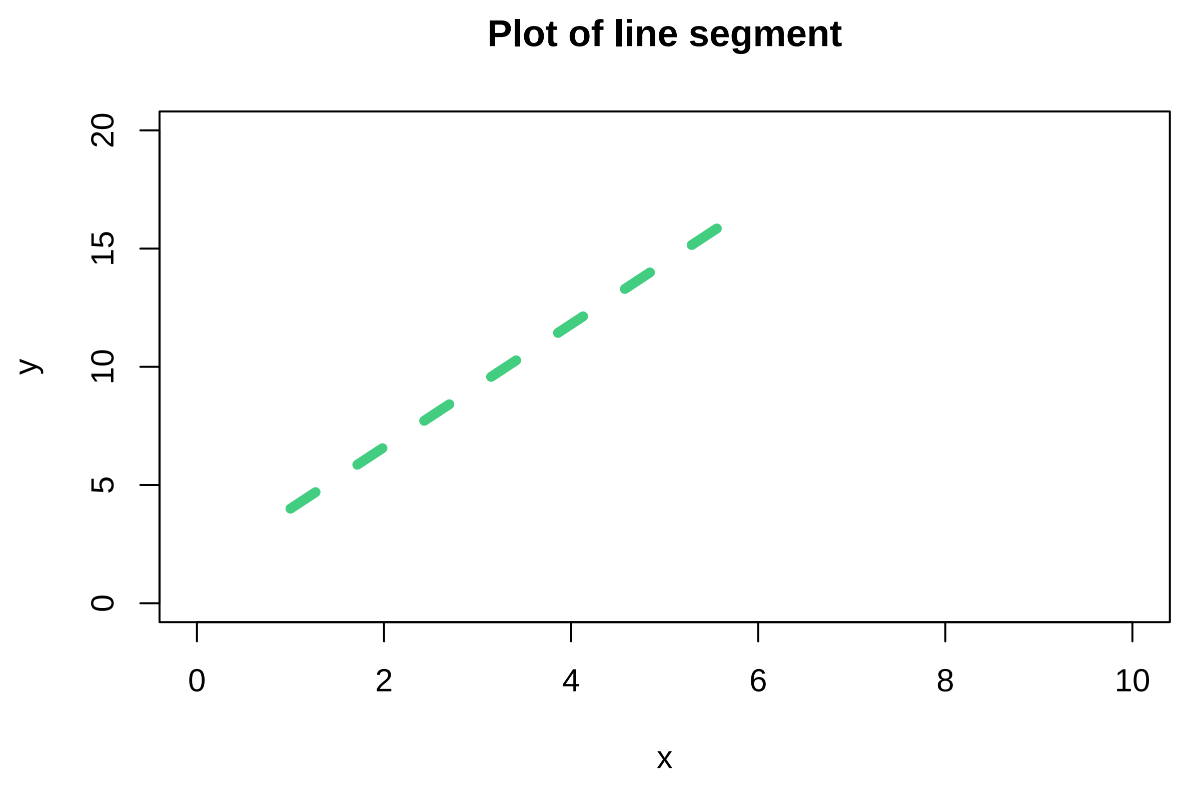
Notice that we have to specify that we want to draw a line by using the character string "l".

```r
plot(
    x = c(1, 6),
    y = c(4, 17),
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of line segment",
    xlab = "x",
    ylab = "y",
    type = "l"
)
```



We can adjust the line type, width, and color of this line segment by using the same optional input arguments that we saw in the previous section with the `segments()` function:

```r
plot(
    x = c(1, 6),
    y = c(4, 17),
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of line segment",
    xlab = "x",
    ylab = "y",
    type = "l",
    lty = "dashed",
    lwd = 5,
    col = "seagreen3"
)
```



In fact, we can actually use the `plot()` function to draw a sequence of connected line segments.

For instance, suppose we want to draw a second line segment starting at the ending point of the first line segment.

That is, the first line segment ended at the point (6, 17), and we want to start our second line segment at this point.

It gets confusing saying things like "ending point of the first line segment", so I'm just going to label the points as first, second, and third.

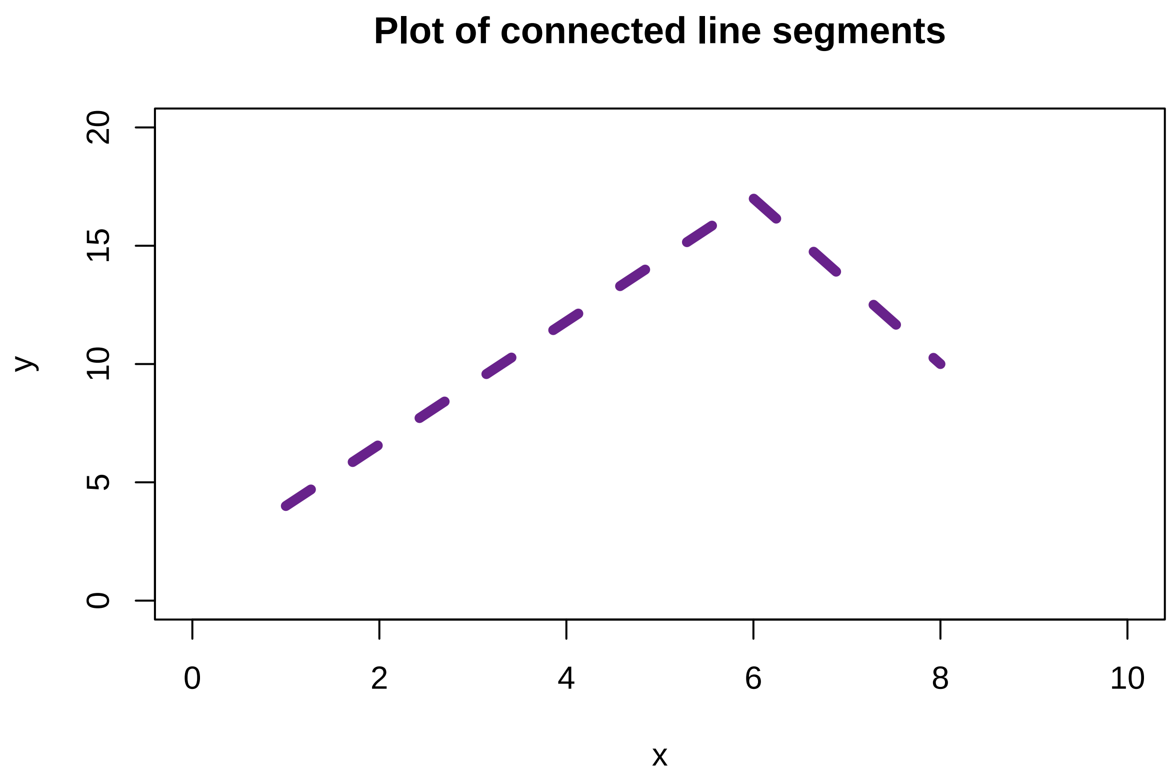
Then the first line segment connects the first point with the second point, and the second line segment connects the second point with the third point.

Let's make a table of these points:

Points	x	y
First	1	4
Second	6	17
Third	8	10

Let's draw these two connected line segments by using vectors for the x- and y-coordinates with the `plot()` function:

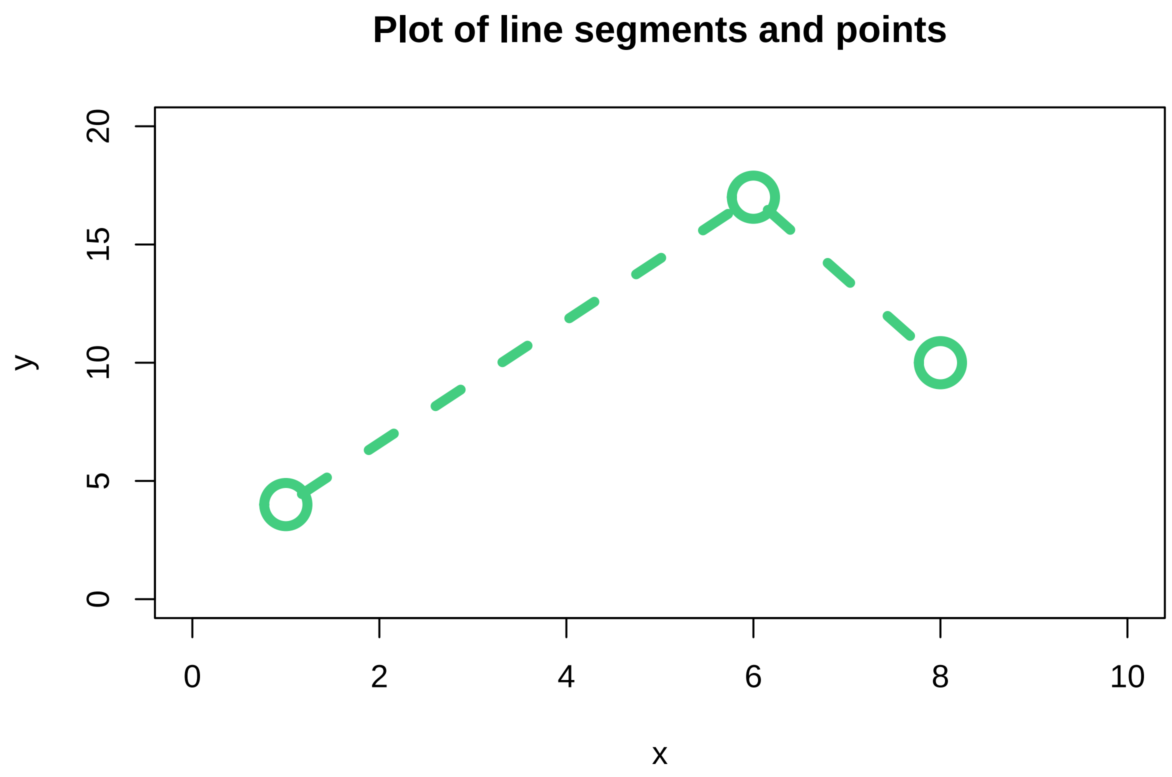
```
plot(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of connected line segments",
    xlab = "x",
    ylab = "y",
    type = "l",
    lty = "dashed",
    lwd = 5,
    col = "darkorchid4"
)
```



If we use `type = "b"` instead of `type = "l"`then `plot()` will draw both points and lines:

```
plot(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of line segments and points",
    xlab = "x",
    ylab = "y",
    type = "b",
    cex = 3,
    lty = "dashed",
    lwd = 5,
```

```r
    col = "seagreen3"
)
```



So that's how to use the `plot()` function to draw line segments.

Now let's see how use the `lines()` function to draw a sequence of connected line segments.

Exercise 4.3: Annual profit line graph

The annual profit for WiDgT for the years 2016 - 2019 is:

Year	Annual Profit (M)
2016	1.3
2017	1.6
2018	2.1
2019	2.2

Create a connected graph of the annual profits using the `plot()` function. Include both the line as well as the points.

Solution

```r
# Type your answer here
```

Section 4: The `lines()` Function

Main Idea: *We can draw a sequence of connected line segments by using the `lines()` function*

In this section, we'll learn how to draw a sequence of connected line segments by using the `lines()` function.

So far, we've seen how to draw line segments by using the `segments()` function and also by using the `plot()` function.

There's another function to draw line segments, called `lines()`.

Now, the `lines()` function cannot create a plotting region, so there has to be a pre-existing plot in order to use this function.

We'll continue with our example: the first point is (1, 4), the second point is (6, 17), and the third point is (8, 10).

We can make a table of these points:

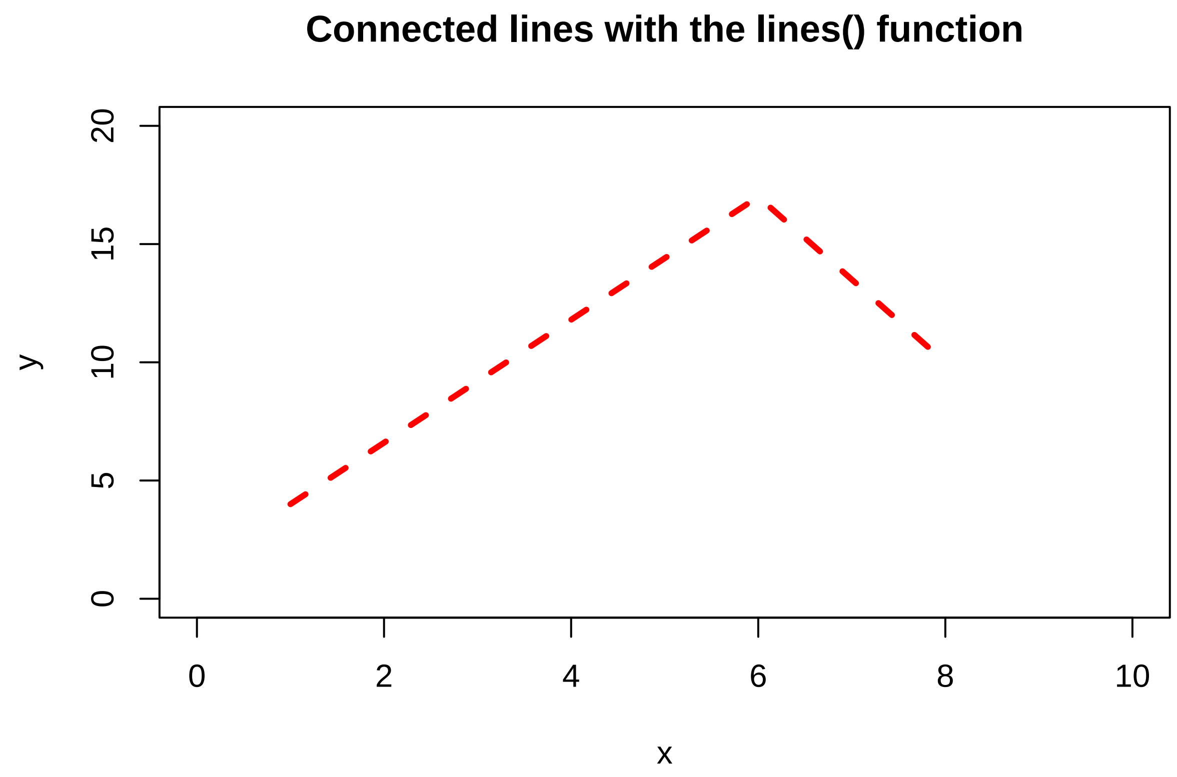
Points	x	y
First	1	4
Second	6	17
Third	8	10

Looking at the table, you can see that the x coordinates of the points are 1, 6, and 8, while the y coordinates are 4, 17, and 10.

We'll employ our standard strategy of using a vector to hold all the x-coordinates and another vector to hold all the y-coordinates.

Now we can draw a sequence of connected line segments by using the `lines()` function:

```r
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Connected lines with the lines() function",
    xlab = "x",
    ylab = "y"
)

lines(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    lty = "dashed",
    lwd = 3,
    col = "red"
)
```

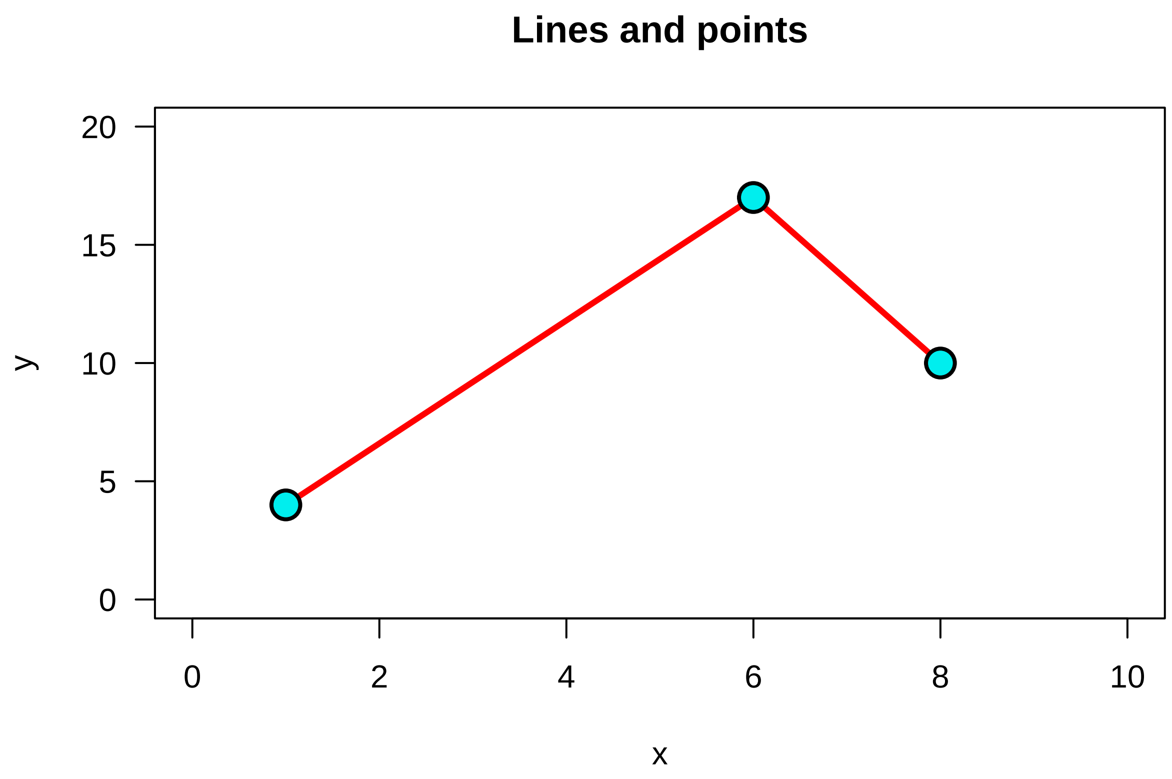


The advantage of the `lines()` function is that we can use it to create different sets of connected line segments, each with its own line type, width, and color.

This is similar to using the `points()` function to draw different kinds of points on a graph.

Once we've created this graph, we can layer some points on top of it by using the `points()` function:

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Lines and points",
    xlab = "x",
    ylab = "y",
    las = 1
)

lines(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    lty = "solid",
    lwd = 3,
    col = "red"
)


points(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    pch = 21,
    cex = 2,
    col = "black",
    lwd = 2,
    bg = "cyan2"
)
```



So that's how to draw a sequence of connected line segments by using the `lines()` function.

Now let's see how to draw polygons.

Exercise 4.4: Using the `lines()` function

Create an empty plot with no data, with x ranging from 0 to 20 and y ranging from 0 to 10.

Then draw two connected line segments:

- The first line segment starts at the point (3, 7) and ends at the point (12, 4).
- The second line segment starts at the point (12, 4) and ends at the point (16, 9).

Use a solid line with a width of 2 and a color of "aquamarine3".

Solution

```
# Type your answer here
```

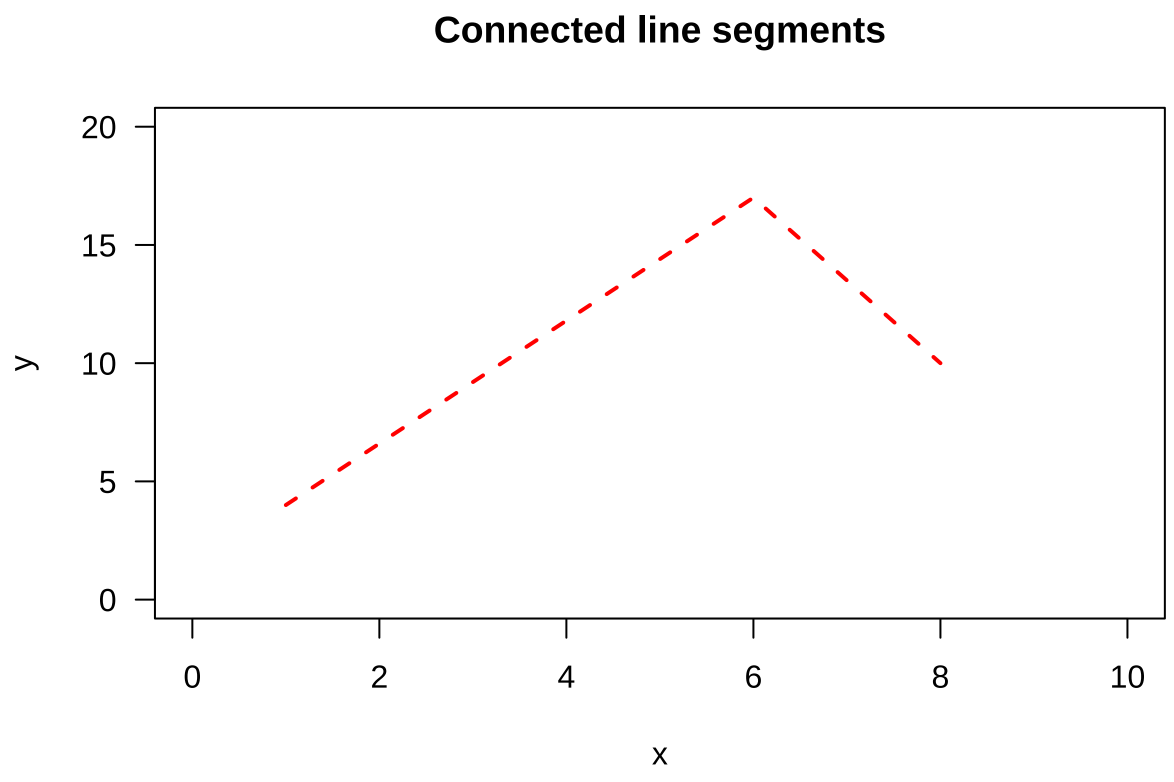
Section 5: The `polygon()` Function

Main Idea: *We can draw polygonal shapes by using the `polygon()` function.*

In this section, we'll learn how to draw polygons by using the `polygon()` function.

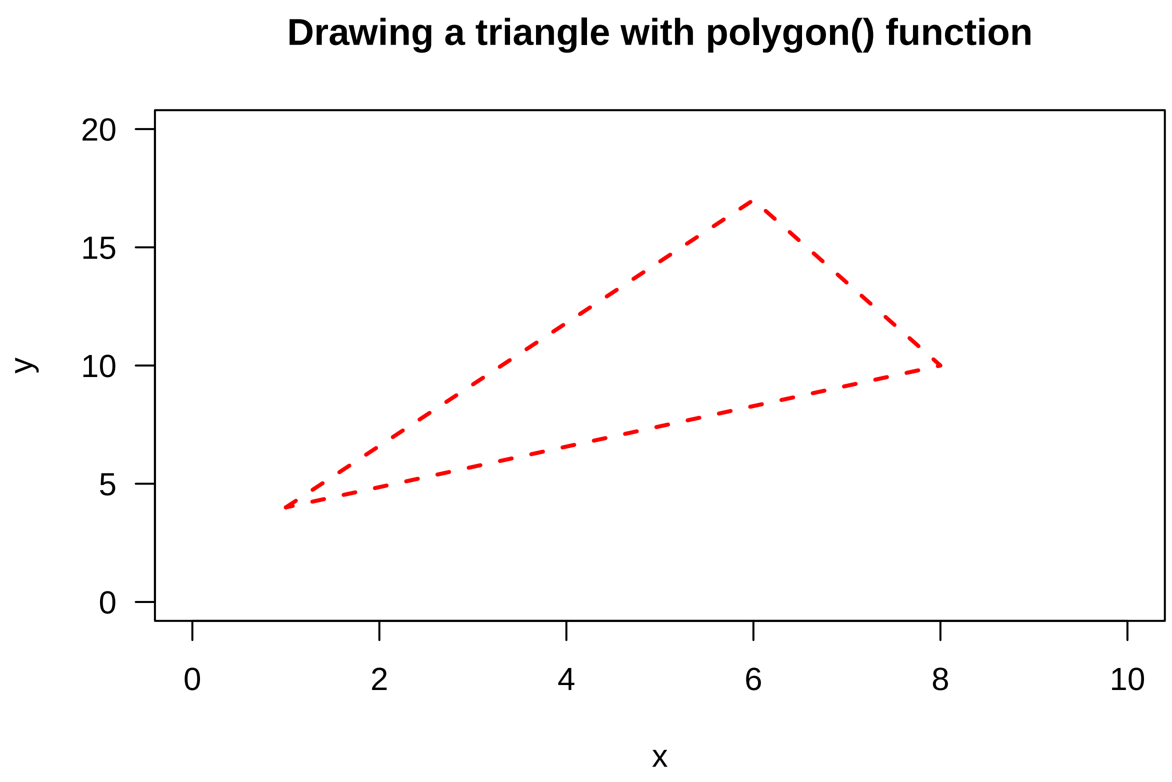
Let's go back to the shape that we created using the `lines()` function:

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Connected line segments",
    xlab = "x",
    ylab = "y",
    las = 1
)

lines(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    lty = "dashed",
    lwd = 2,
    col = "red"
)
```



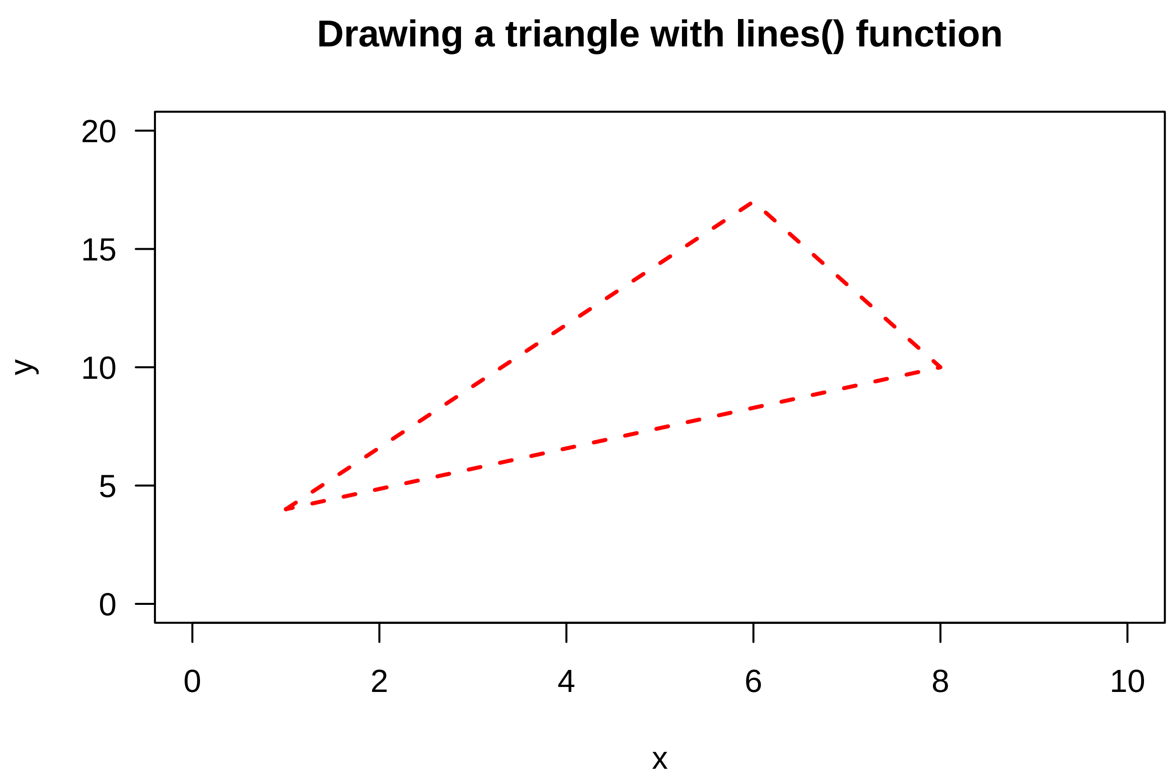
If we use the `polygon()` function instead, it will connect the last point with the first point:

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Drawing a triangle with polygon() function",
    xlab = "x",
    ylab = "y",
    las = 1
)

polygon(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    lty = "dashed",
    lwd = 2,
    border = "red"
)
```



If we want to draw this triangle using the `lines()` function, we have to explicitly connect the last point to the first point:

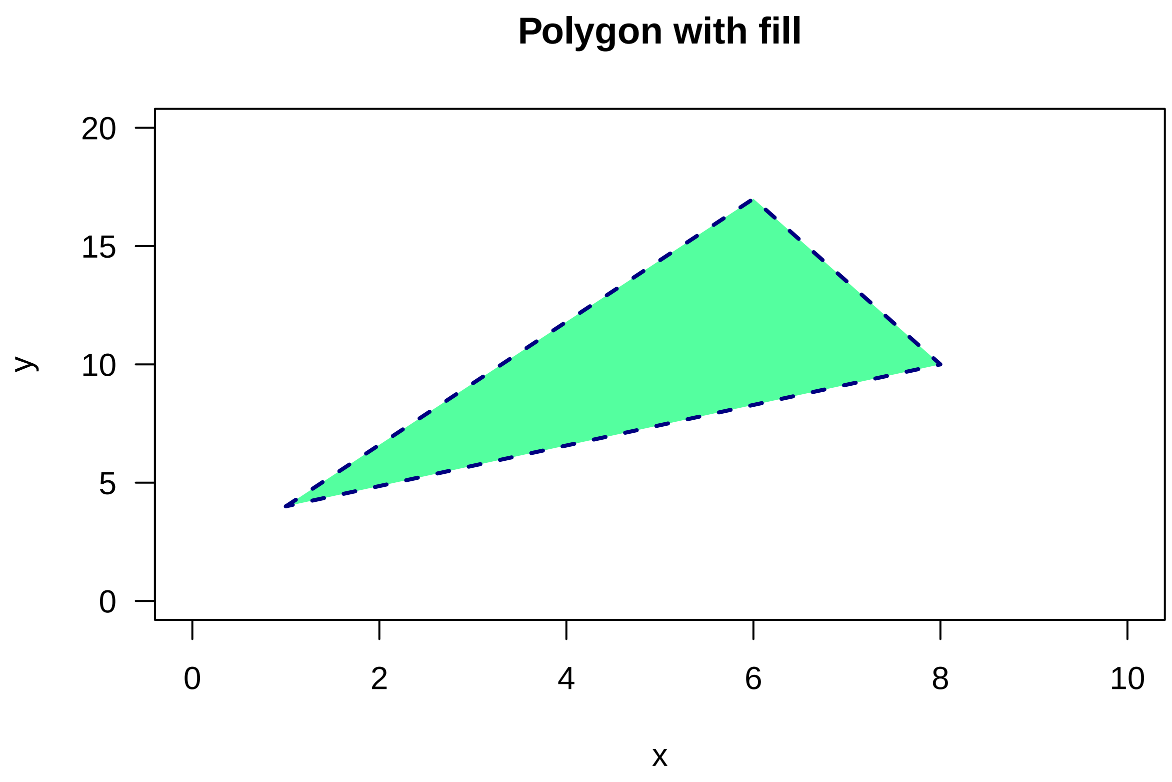
```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Drawing a triangle with lines() function",
    xlab = "x",
    ylab = "y",
    las = 1
)

lines(
    x = c(1, 6, 8, 1),
    y = c(4, 17, 10, 4),
    lty = "dashed",
    lwd = 2,
    col = "red"
)
```



There's another amazing feature of the `polygon()` function – it can fill in the interior of the polygon with a color.

With the `polygon()` function, we specify the color of the line connecting the points by using the `border` option, while the `col` option will actually fill in the polygonal region:

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Polygon with fill",
    xlab = "x",
    ylab = "y",
    las = 1
)

polygon(
    x = c(1, 6, 8),
    y = c(4, 17, 10),
    lty = "dashed",
    lwd = 2,
    border = "navy",
    col = "seagreen1"
)
```



So that's how to draw polygons by using the `polygon()` function.

Now let's review what we've learned in this module.

Exercise 4.5: 3-4-5 Right Triangle

Create an empty plot with no data, where the x-axis ranges from 0 to 10 and the y-axis ranges from 0 to 7. Create a 3-4-5 right triangle using the `polygon()` function, and specify a border color and a fill color. Hint: start at the point (3, 5), then go down 3 units, then go right for 4 units.

Solution

```
# Type your answer here
```

Module Review

In this module, we learned how to draw line segments and polygons.

- In Section 1, we saw how to draw a line segment by using the `segments()` function.
- In Section 2, we learned how to modify the line type, width, and color for a line segment.
- In Section 3, we saw how to use the `plot()` function to draw line segments.
- In Section 4, we learned how to draw a sequence of connected line segments by using the `lines()` function.
- In Section 5, we learned how to draw polygons by using the `polygon()` function.

Now that you've completed this module, you should be able to:

- Draw a line segment on a pre-existing plotting region by using the `segments()` function.
- Adjust the line type, line width, and color of a line segment.
- Draw line segments by using the `plot()` function.
- Draw a connected sequence of line segments by using the `lines()` function.
- Create a polygonal shape by using the `polygon()` function.

In this module, we met three new built-in R functions:

- `segments()`
- `lines()`
- `polygon()`

All right! That's it for Module 4: Lines and Polygons.

Now let's move on to Module 5: Curves.

Solutions to the Exercises

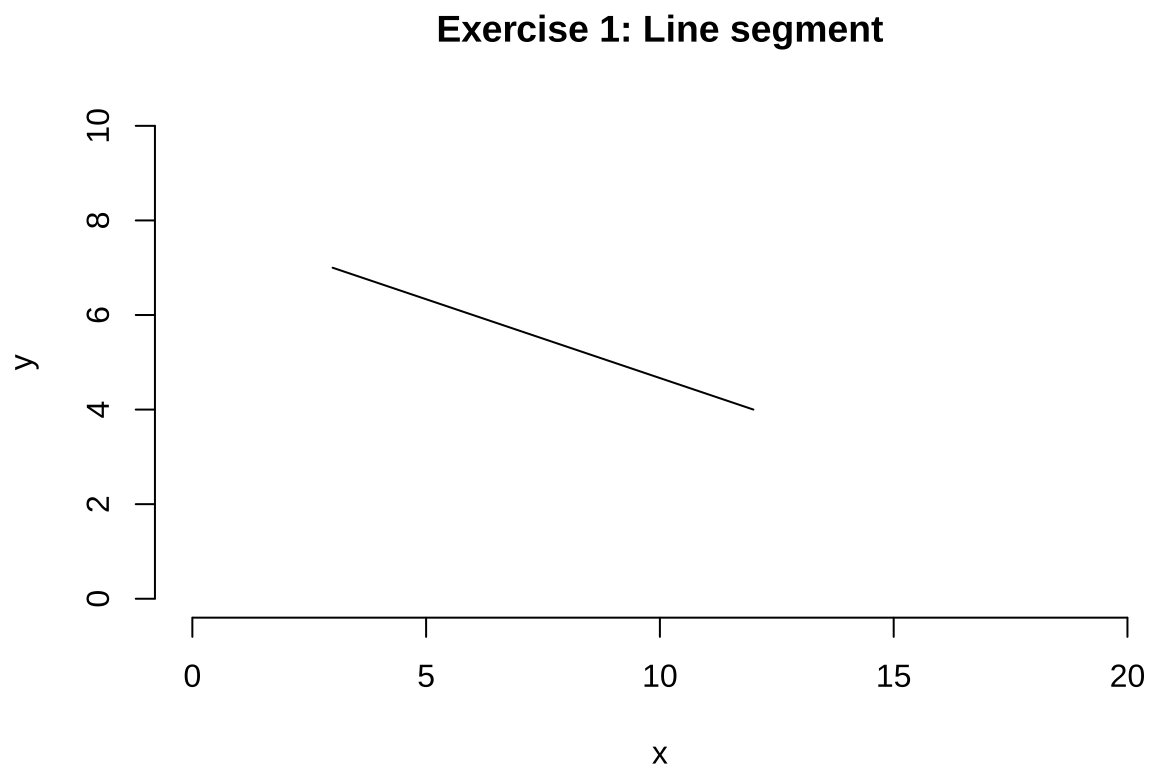
Exercise 4.1: Draw a line segment

Create an empty plot with no data, with x ranging from 0 to 20 and y ranging from 0 to 510.

Then draw a line segment starting at the point (3, 7) and ending at the point (12, 4).

Solution

```r
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 20),
    ylim = c(0, 10),
    main = "Exercise 1: Line segment",
    xlab = "x",
    ylab = "y",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 3,
    y0 = 7,
    x1 = 12,
    y1 = 4
)
```



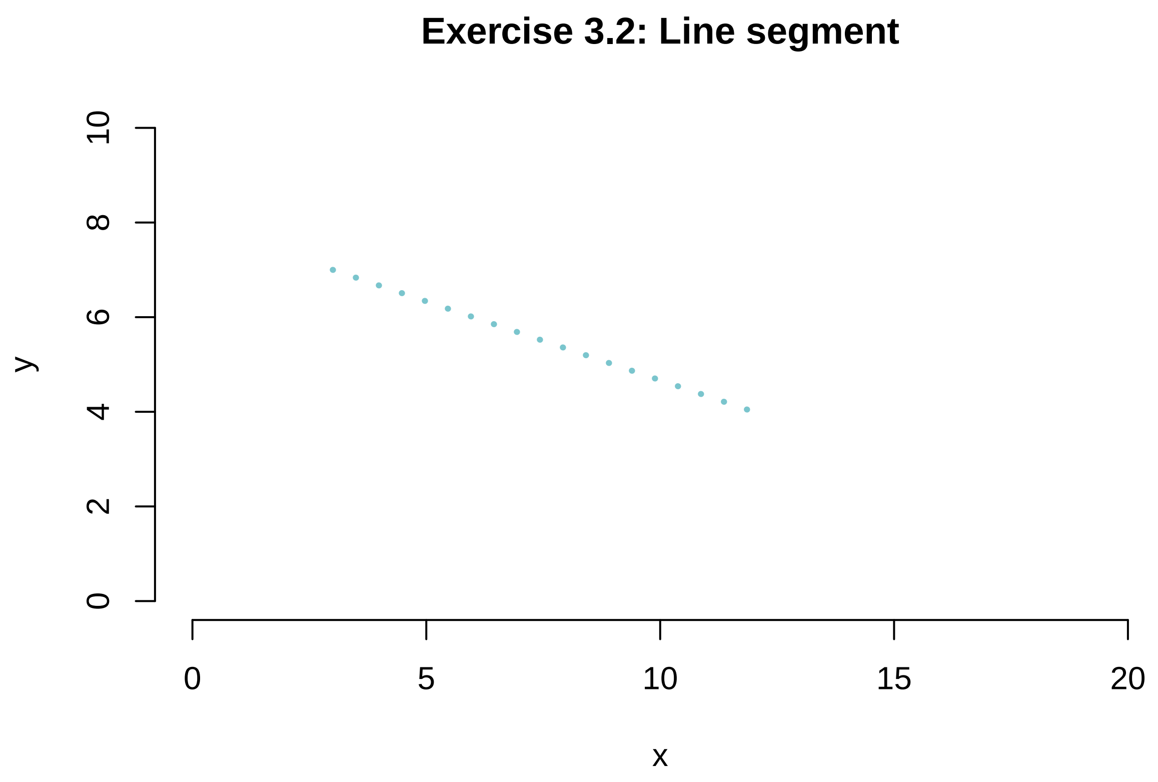
Exercise 4.2: Line type, width, and color

Create an empty plot with no data, with x ranging from 0 to 20 and y ranging from 0 to 10.

Then draw a dotted line segment starting at the point (3, 7) and ending at the point (12, 4), using a line width of 3 and a color of "cadetblue3".

Solution

```
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 20),
    ylim = c(0, 10),
    main = "Exercise 3.2: Line segment",
    xlab = "x",
    ylab = "y",
    bty = "n"
)

# Now let's draw the line segment:

segments(
    x0 = 3,
    y0 = 7,
    x1 = 12,
    y1 = 4,
    lty = "dotted",
    lwd = 3,
    col = "cadetblue3"
)
```



Exercise 4.3: Annual profit line graph

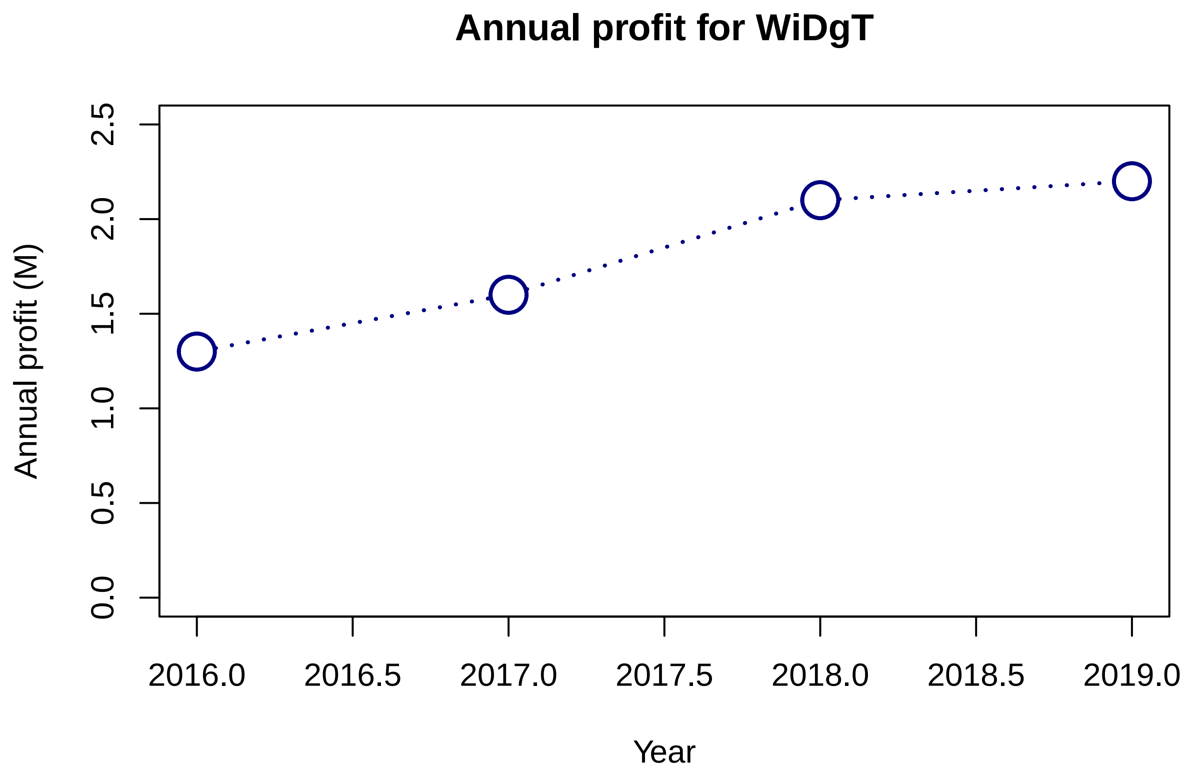
The annual profit for WiDgT for the years 2016 - 2019 is:

Year	Annual Profit (M)
2016	1.3
2017	1.6
2018	2.1
2019	2.2

Create a connected graph of the annual profits using the `plot()` function. Include both the line as well as the points.

Solution

```r
plot(
    x = c(2016, 2017, 2018, 2019),
    y = c(1.3, 1.6, 2.1, 2.2),
    xlim = c(2016, 2019),
    ylim = c(0, 2.5),
    main = "Annual profit for WiDgT",
    xlab = "Year",
    ylab = "Annual profit (M)",
    col = "navy",
    cex = 2.5,
    lty = "dotted",
    lwd = 2,
    type = "b"
)
```



Exercise 4.4: Using the `lines()` function

Create an empty plot with no data, with x ranging from 0 to 20 and y ranging from 0 to 10.

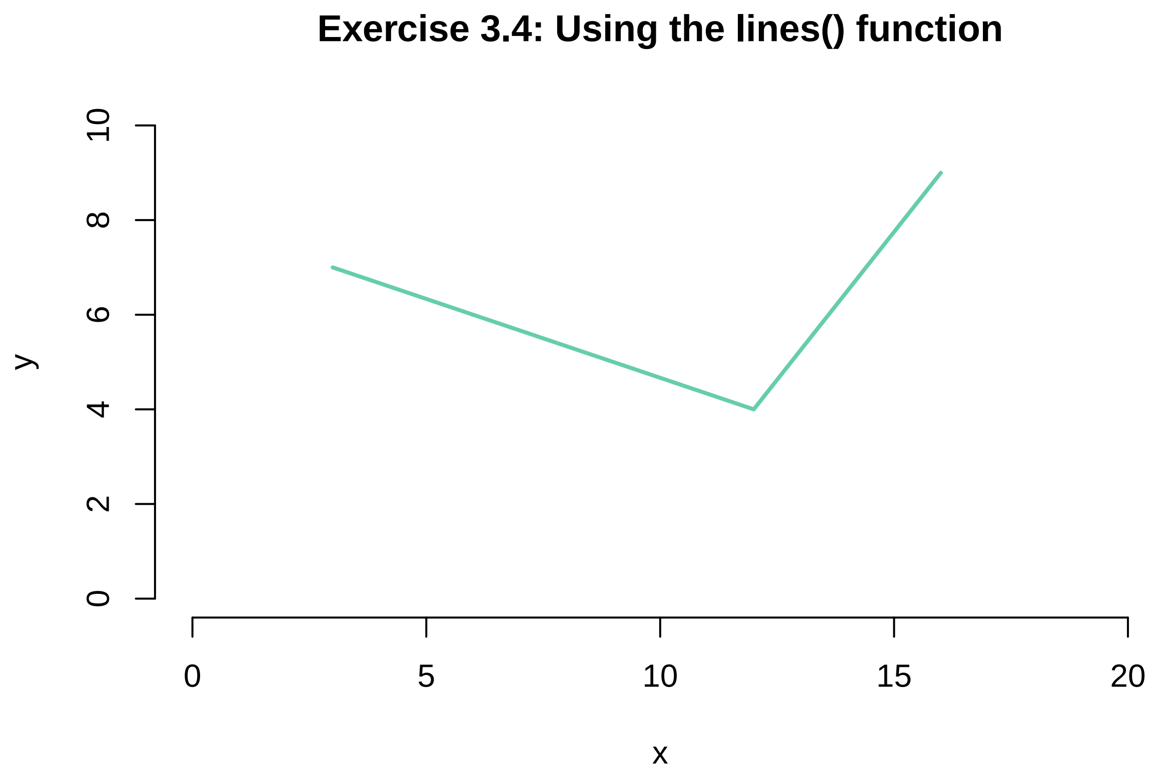
Then draw two connected line segments:

- The first line segment starts at the point (3, 7) and ends at the point (12, 4).
- The second line segment starts at the point (12, 4) and ends at the point (16, 9).

Use a solid line with a width of 2 and a color of “aquamarine3”.

Solution

```
# First, let's create the empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 20),
    ylim = c(0, 10),
    main = "Exercise 3.4: Using the lines() function",
    xlab = "x",
    ylab = "y",
    bty = "n"
)

# Now let's draw the line segments using
# the lines() function:

lines(
    x = c(3, 12, 16),
    y = c(7, 4, 9),
    lty = "solid",
    lwd = 2,
    col = "aquamarine3"
)
```



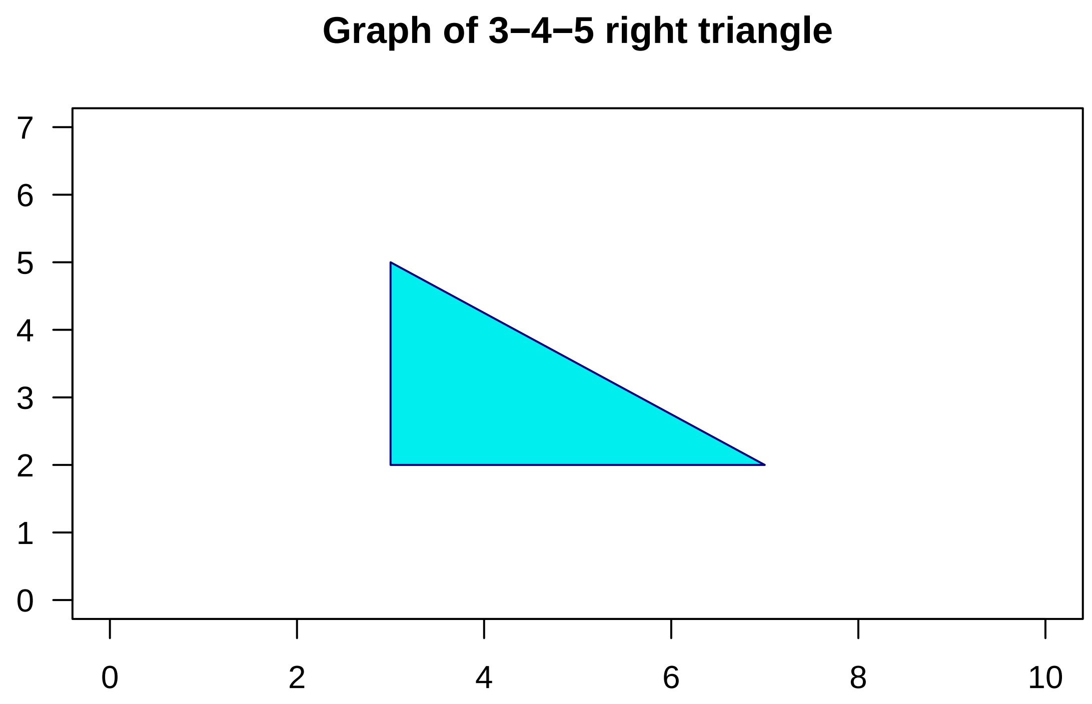
Exercise 4.5: 3-4-5 Right Triangle

Create an empty plot with no data, where the x-axis ranges from 0 to 10 and the y-axis ranges from 0 to 7. Create a 3-4-5 right triangle using the `polygon()` function, and specify a border color and a fill color. Hint: start at the point (3, 5), then go down 3 units, then go right for 4 units.

Solution

Here's my solution:

```r
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 7),
    main = "Graph of 3-4-5 right triangle",
    xlab = "",
    ylab = "",
    las = 1
)

polygon(
    c(3, 3, 7),
    c(5, 2, 2),
    border = "navy",
    col = "cyan2"
)
```


Graph of 3−4−5 right triangle