Week 1 Module 3: Points

CSCI E-5a: Programming in R

Let's clear the global computing environment:

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

Module Preview

Hello! And welcome to Module 3: Points.

In this module, we'll learn how to work with points.

- In Section 1, we'll see how to add points to a pre-existing graphics plot.
- In Section 2, we'll find out how to annotate a plot with text.
- In Section 3, we'll learn how to adjust the size of a point.
- In Section 4, we'll see how to modify the color of a point.
- In Section 5, we'll learn how to modify the shape of a point.
- In Section 6, we'll learn how to plot multiple points by using a single function call.

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

- Add points to a pre-existing plotting region by using the `points()` function
- Annotate a graph with text by using the `text()` function
- Adjust the size of a point by using the `cex` option
- Modify the color of a point by using the `col` option
- Control the shape of a point by using the `pch` option
- Plot multiple points by using a single call of the `plot()` function

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

- `points()`
- `text()`

All right! Let's get started by seeing how to add points to a pre-existing graphics plot.

Section 1: Adding Points to a Plot

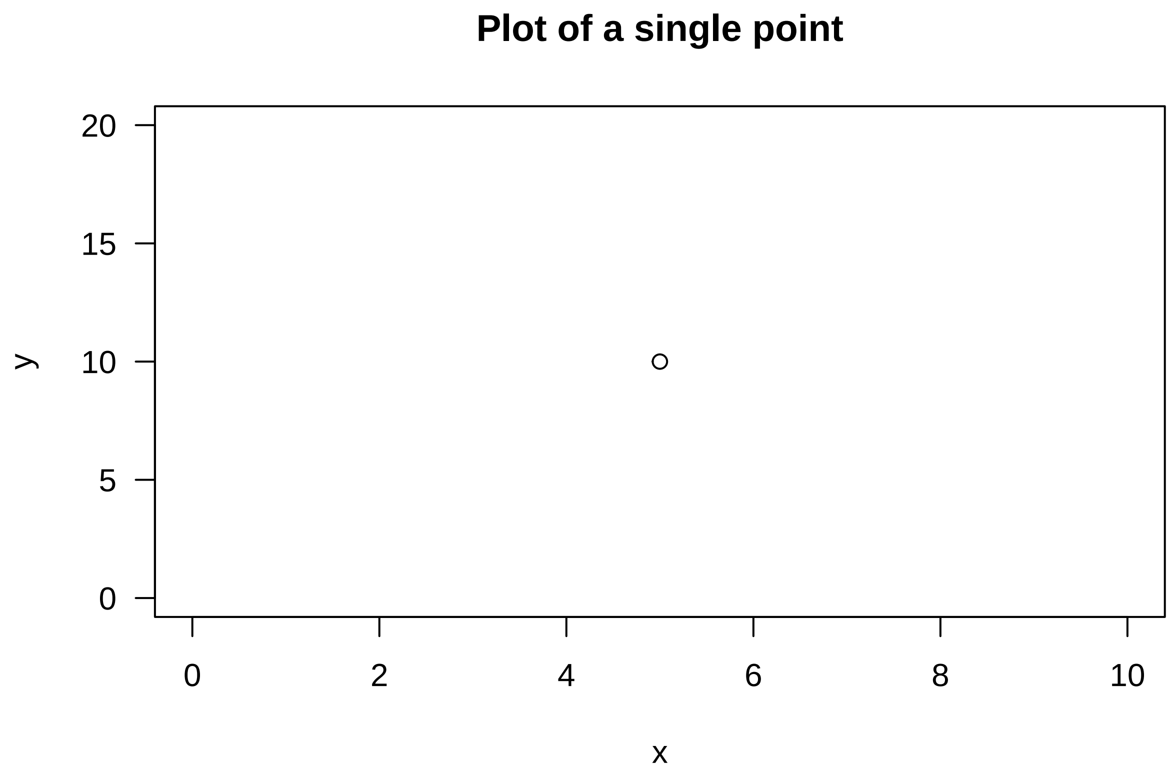
Main Idea: *We can add points to a pre-existing graphics plot.*

In this section, we'll see how to add points to a pre-existing graphics plot.

Once we've created a plot, we can draw points on it by using the `points()` function.

Let's recall the graph of a single point that we created in Module 1:

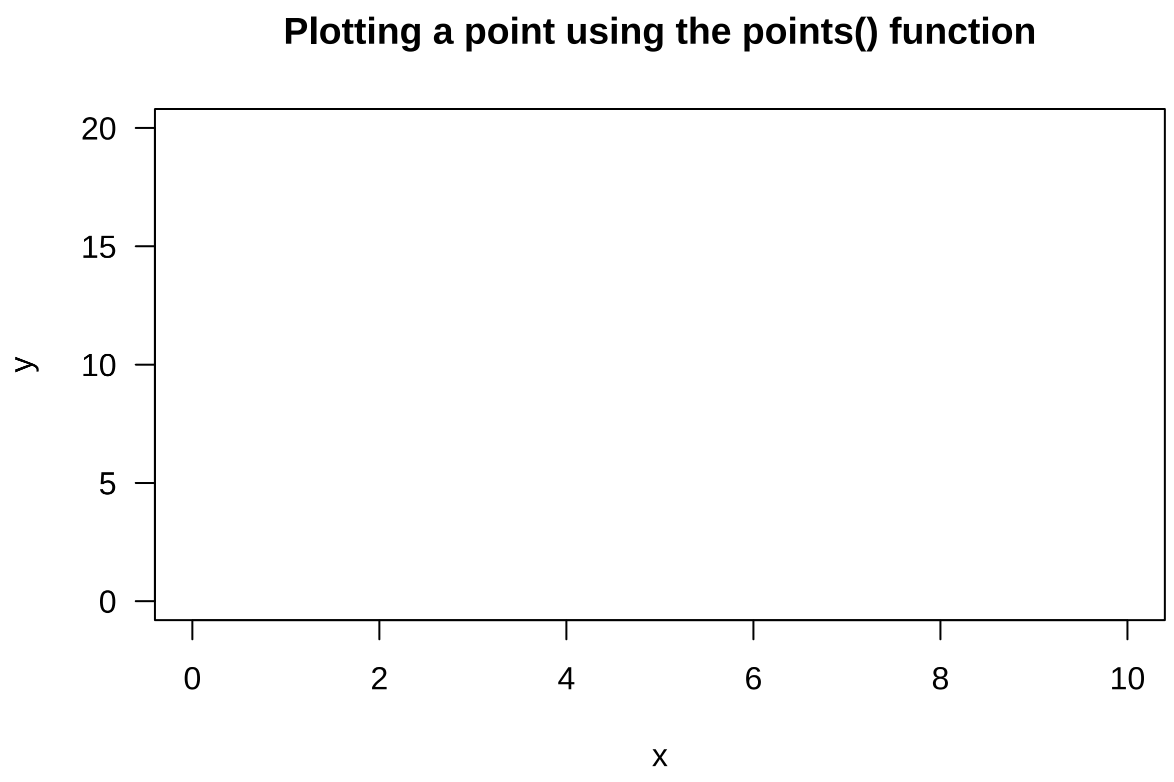
```r
plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1
)
```



There's an alternative approach, in which we first create an empty plot with no data, and then add in the point by using the `points()` function.

First, let's recall how to create an empty plot:

```r
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plotting a point using the points() function",
    xlab = "x",
    ylab = "y",
    las = 1
)
```



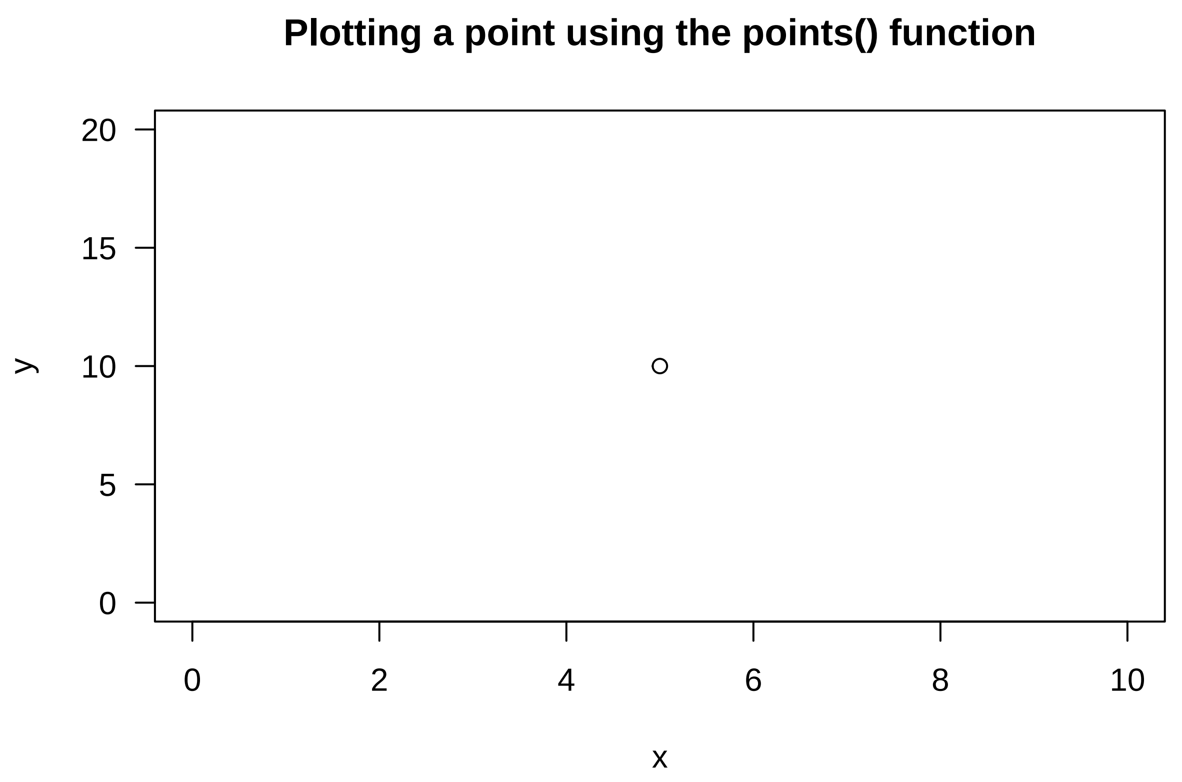
Now we'll create the empty plot, and then add in the point by using the `points()` function.

The `points()` function takes two input arguments:

- The x coordinate, named x.
- The y coordinate, named y.

We can add a point at a specific location to the pre-existing plotting region by calling the `points()` function after we call the `plot()` function, but within the same code chunk:

```r
# First, create an empty plot with no data

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plotting a point using the points() function",
    xlab = "x",
    ylab = "y",
    las = 1
)


# Draw a single point at the location (5, 10):

points(
    x = 5,
    y = 10
)
```



In order to use the `points()` function, there **must** be a pre-existing plotting region.

That's why we had to make the empty plotting region first, so that we could then use the `points()` function.

On the other hand, the `plot()` function always creates a new plotting region, as well as plotting the point.

We can use the `points()` function repeatedly with a single graph, so we can create visualizations with points that have different colors, sizes, and shapes.

So that's how we can add points to a pre-existing graphics plot.

Now let's see how to add text to a graphics plot.

Exercise 3.1: Graphing a single point

First, create an empty plotting region with no data, where the x-values range from 0 to 10, and the y-values range from 0 to 8$.

Then use the `points()` function to draw a single point located at $x = 7$ and $y = 3$.

Solution

Section 2: Text

Main Idea: *We can annotate a plot with text.*

In this section, we'll find out how to annotate a plot with text.

Now, sometimes we want to write text on the graphics display.

For example, with our simple graph it would be nice to actually indicate the coordinates of the point.

We can use the built-in R function `text()` to write text annotations on the plotting region.

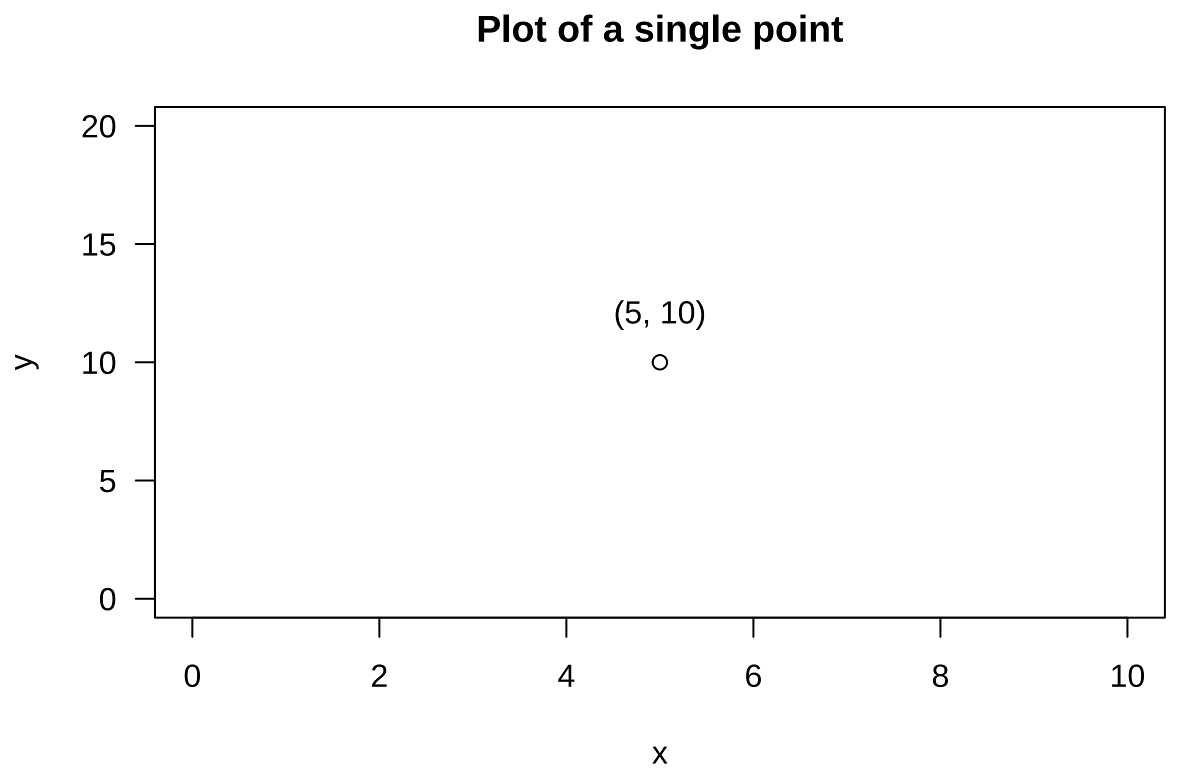
The `text()` function takes 3 input arguments:

- The first input argument, named `x`, determines the x-coordinate for the location of the text, and is specified by a single numeric value.
- The second input argument, named `y`, determines the y-coordinate for the location of the text, and is specified by a single numeric value.
- The third input argument, named `labels`, determines the actual text, and is specified by a single character string.

The `text()` function also has an option named 'pos', which allows us to control the justification of the text.

Let's redraw our simple graph, but this time indicating the coordinates of the point.

```r
# First we'll draw a single point using the plot() function

plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1
)

# Next we'll annotate this point with text

text(
    x = 5,
    y = 12,
    labels = "(5, 10)"
)
```



How did I choose the coordinates for the text?

I wanted the text to be centered over the point, so that meant that the x coordinate was 5.

To determine the y coordinate, I just used trial-and-error to find a value that looked good.

So that's how to annotate a plot with text.

Now let's see how to adjust the size of a point.

Exercise 3.2: Annotating with text

Draw a graph with a single point located at $x = 7$ and $y = 3$.

Set the x-values for the plotting region to range from 0 to 10, and the y-values to range from 0 to 8.

Then annotate this point with its coordinates using the `text()` function.

Solution

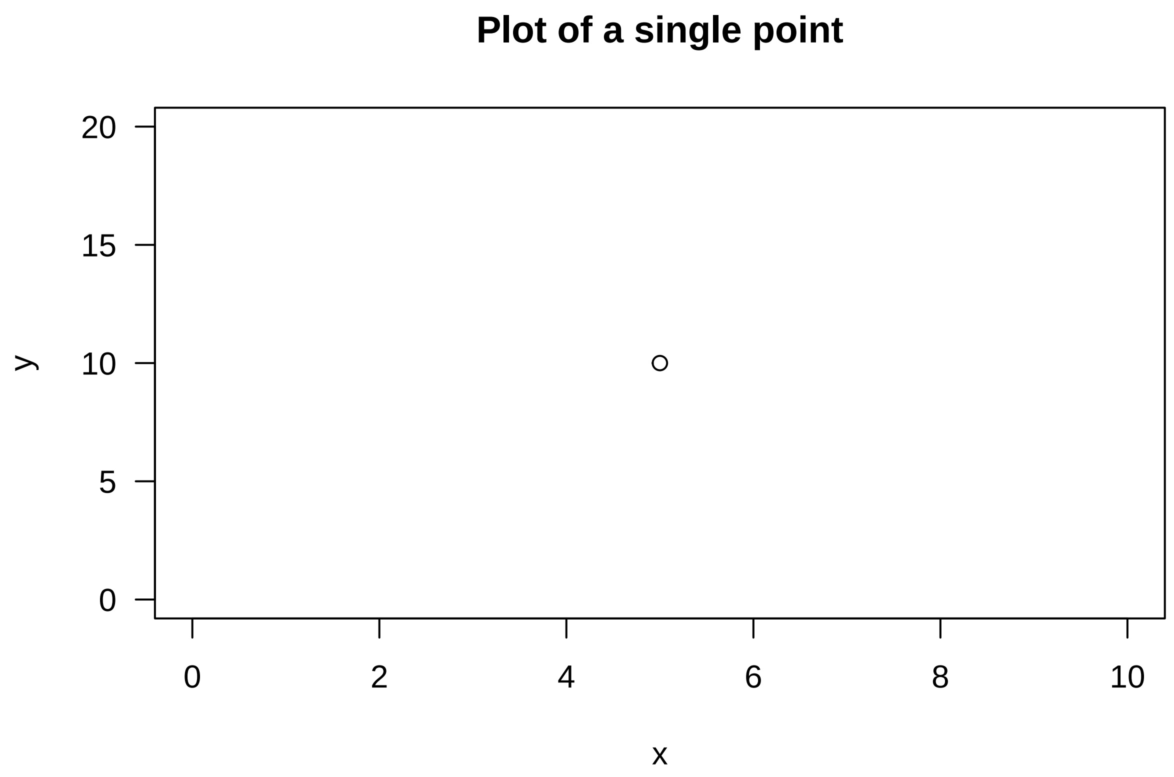
Section 3: Point size

Main Idea: *We can adjust the size of a point*

In this section, we'll see how to adjust the size of a point by using the `cex` option.

Let's return to our basic graph of a single point:

```
plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1
)
```

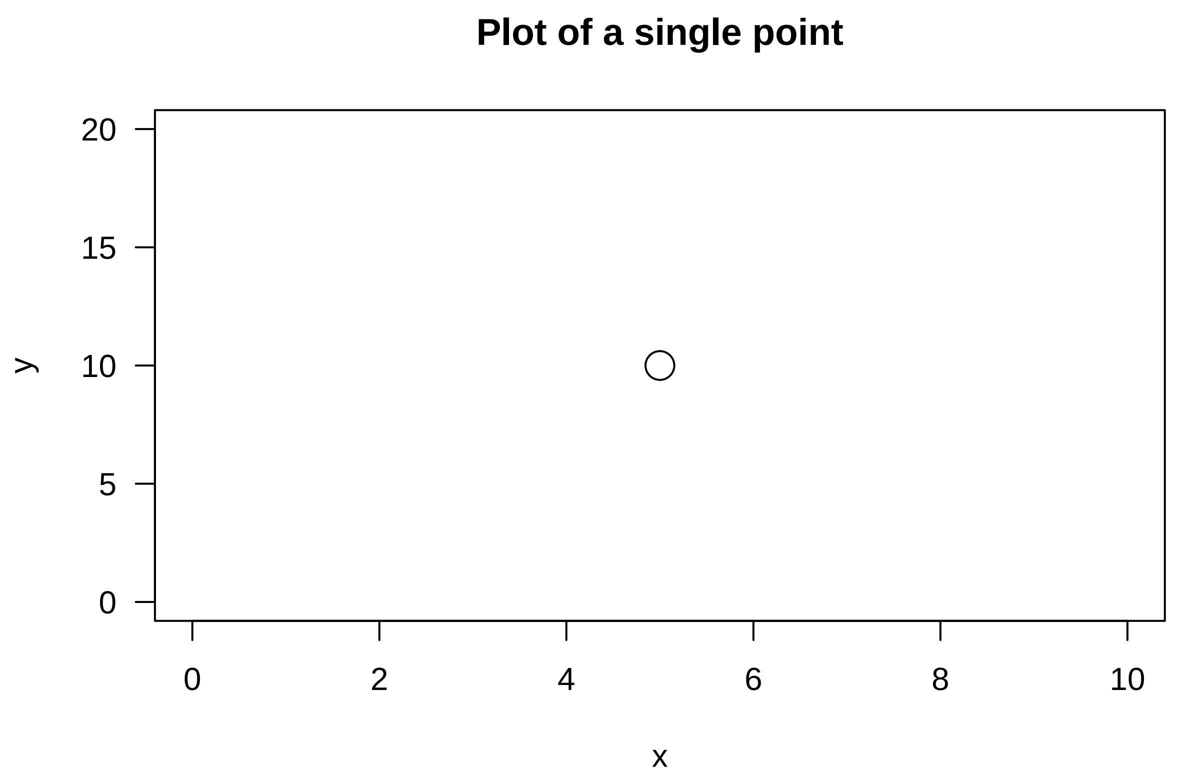


Notice that the point in this graph is very small, and if it were larger we could see it more easily.

We can make the point larger by specifying a *character expansion factor* using the `cex` option.

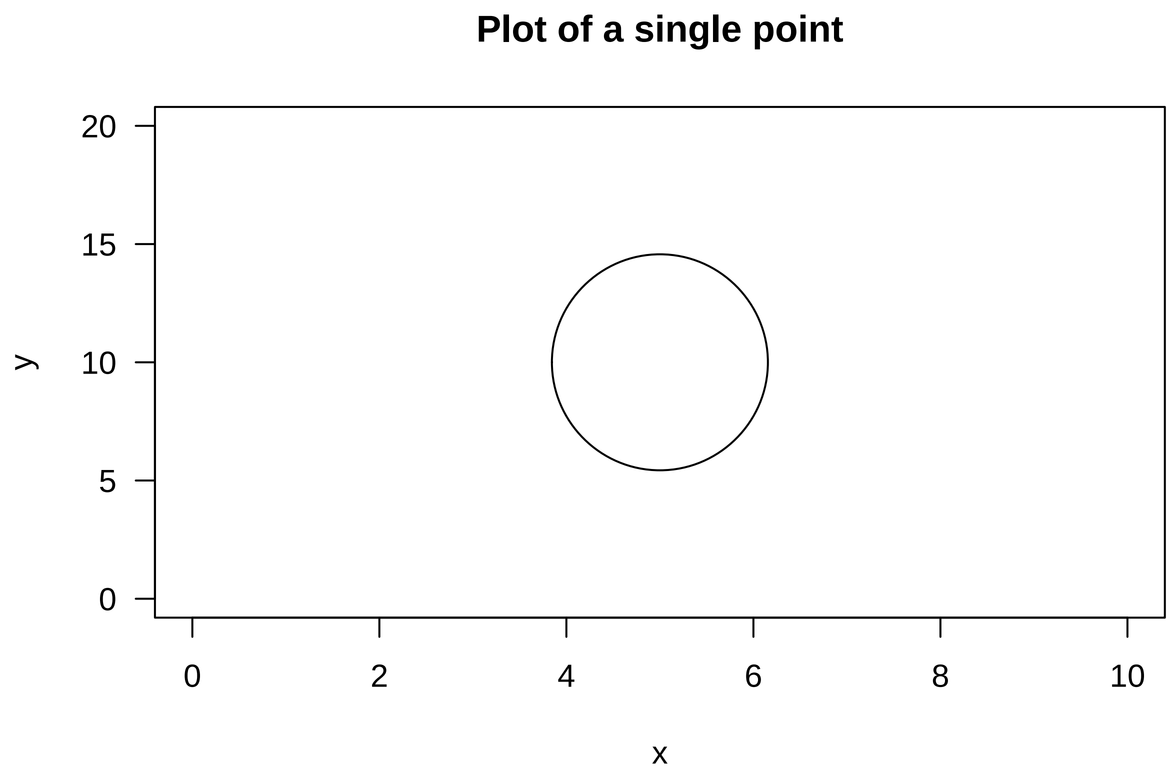
For instance, to make the point twice as large, we have:

```
plot(
    x = 5,
    y = 10,
    main = "Plot of a single point",
    xlim = c(0, 10),
    ylim = c(0, 20),
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 2    # This is where we adjust the size of the point.
)
```



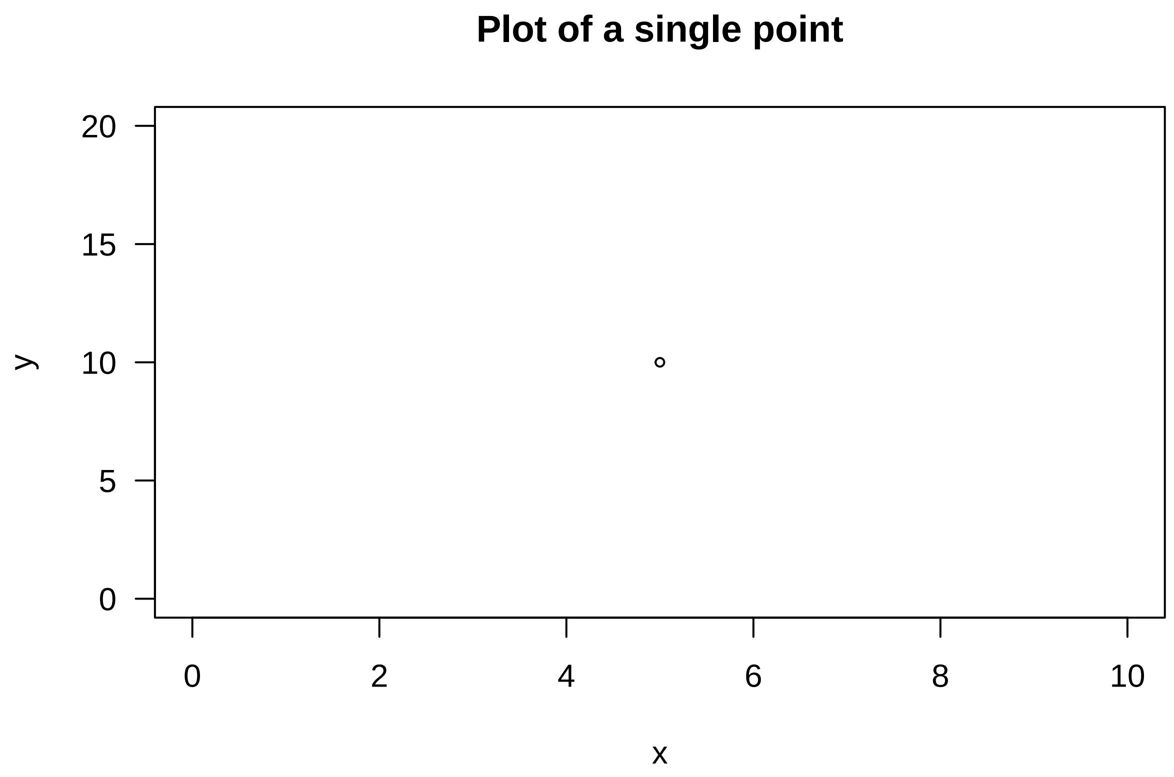
You can see that the point is now twice as large. We can make the point much larger indeed:

```
plot(
    x = 5,
    y = 10,
    main = "Plot of a single point",
    xlim = c(0, 10),
    ylim = c(0, 20),
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 15     # Now we make the point much bigger
)
```



We can also make the point smaller by using a value for `cex` that is less than 1:

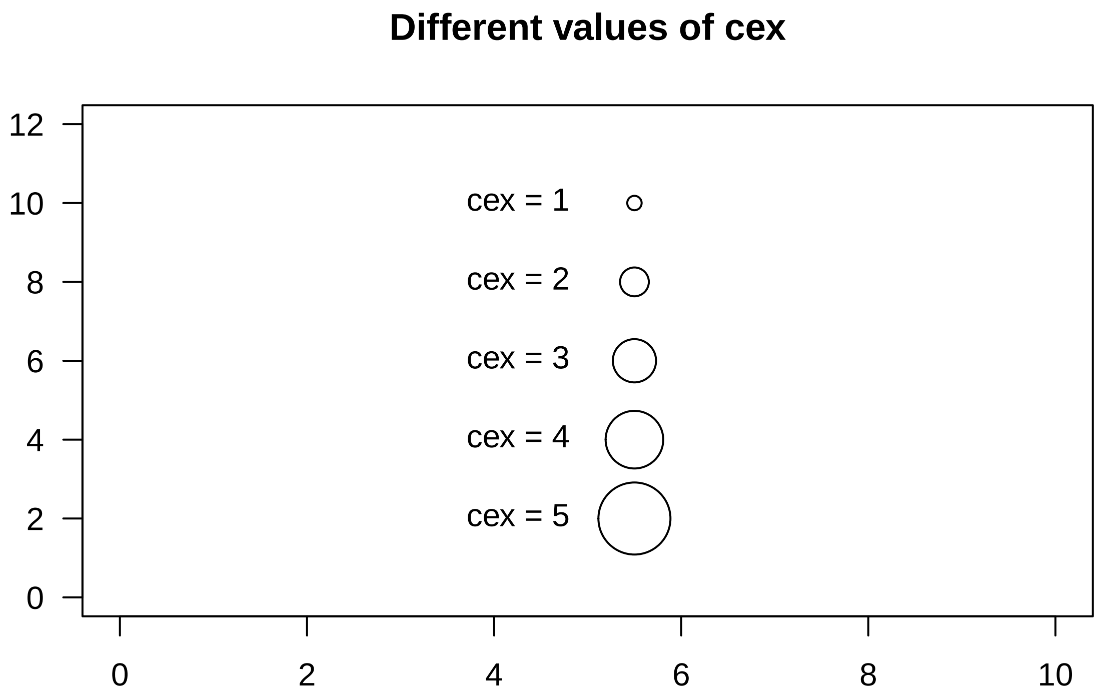
```r
plot(
    x = 5,
    y = 10,
    main = "Plot of a single point",
    xlim = c(0, 10),
    ylim = c(0, 20),
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 0.6     # Now we make the point smaller
)
```



Here is a more complicated image, showing a series of points with different expansion factor values:

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

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 12),
    main = "Different values of cex",
    xlab = "",
    ylab = "",
    las = 1
)

# Now we'll draw a series of points of different sizes,
# and we'll annotate each point with a text caption

# Using pos = 4 in the text() function causes the
# display to be left-justified.

text(
    x = 3.5,
    y = 10,
    "cex = 1",
    pos = 4
)
```

```
points(
    x = 5.5,
    y = 10,
    cex = 1
)


text(
    x = 3.5,
    y = 8,
    "cex = 2",
    pos = 4
)

points(
    x = 5.5,
    y = 8,
    cex = 2
)


text(
    x = 3.5,
    y = 6,
    "cex = 3",
    pos = 4
)

points(
    x = 5.5,
    y = 6,
    cex = 3
)


text(
    x = 3.5,
    y = 4,
    "cex = 4",
    pos = 4
)

points(
    x = 5.5,
    y = 4,
    cex = 4
)


text(
    x = 3.5,
    y = 2,
    "cex = 5",
    pos = 4
)

points(
    x = 5.5,
    y = 2,
    cex = 5
)
```



This example really illustrates the power of first creating an empty plotting region and then drawing the points, because it enables us to adjust the size of each point and annotate it with its own text.

So that's how we can adjust the size of a point by using the `cex` option.

Now let's see how to modify the color of a point.

Exercise 3.3: Adjusting the size of a point

Create a graph of the single point at $x = 7$ and $y = 12$, and use a character expansion factor of 3.5.

Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

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

Section 4: Point color

Main Idea: *We can change the color of a point.*

In this section, we'll see how to modify the color of a point.

R has a built-in set of over 600 color names.

Here are some example color names:

- "red"
- "blue"
- "hotpink"
- "aquamarine2"

Notice that the color names are actually character strings, so they always have to be enclosed in quotes.

Also, all color names consist entirely of lower case letters and possibly numerals.

Color names never contain special characters, and in particular they never contain spaces, so there are no color names of the form "aquamarine 2".

To obtain a full listing of all the available color names in R, download the "R colors" pdf file.

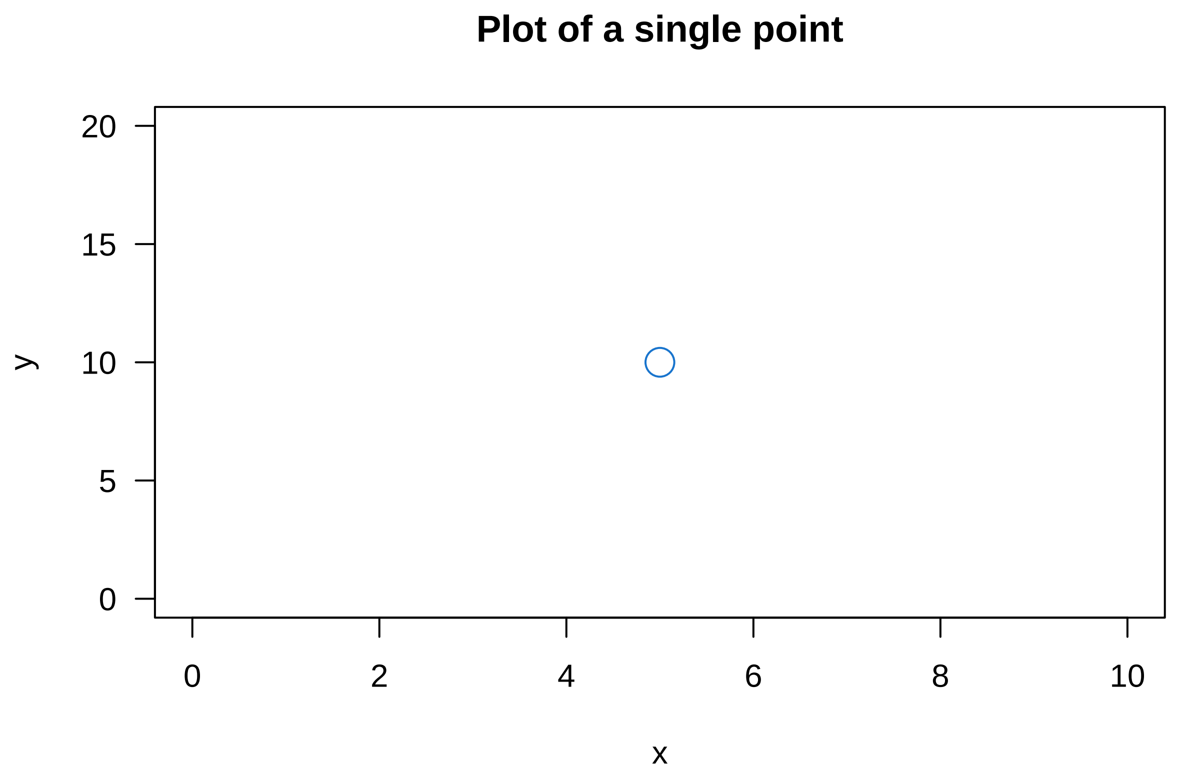
This file consists of color swatches and the associated color name for all the built-in colors in R, so you can quickly search through this to find the color name for the color that you want.

I recommend saving this file in the Documentation subfolder for your course folder that you created back in Week 0, and I think you'll find it very useful for this class and afterwards as well.

To change the color of a point, use the `col` option with a color name.

For instance, let's take our plot of a single point at the location $x = 5$ and $y = 10$, and modify the code so that it has color "dodgerblue3":

```r
plot(
    x = 5,
    y = 10,
    main = "Plot of a single point",
    xlim = c(0, 10),
    ylim = c(0, 20),
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 2,
    col = "dodgerblue3"    # Here is where we specify the color
)
```



So that's how to modify the color of a point by using the `col` option.

Now let's see how to control the shape of a point.

Exercise 3.4: Changing the color of a point

Create a graph of the single point at $x = 7$ and $y = 12$, using a character expansion factor of 3.5 and a color of "hotpink3".

Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

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

Section 5: Point shape

Main Idea: *We can control the shape of a point*

In this section, we'll see how to modify the shape of a point.

The point shape is controlled by the `pch` option, and we can specify 25 different point shapes by using the numbers 1 through 25.

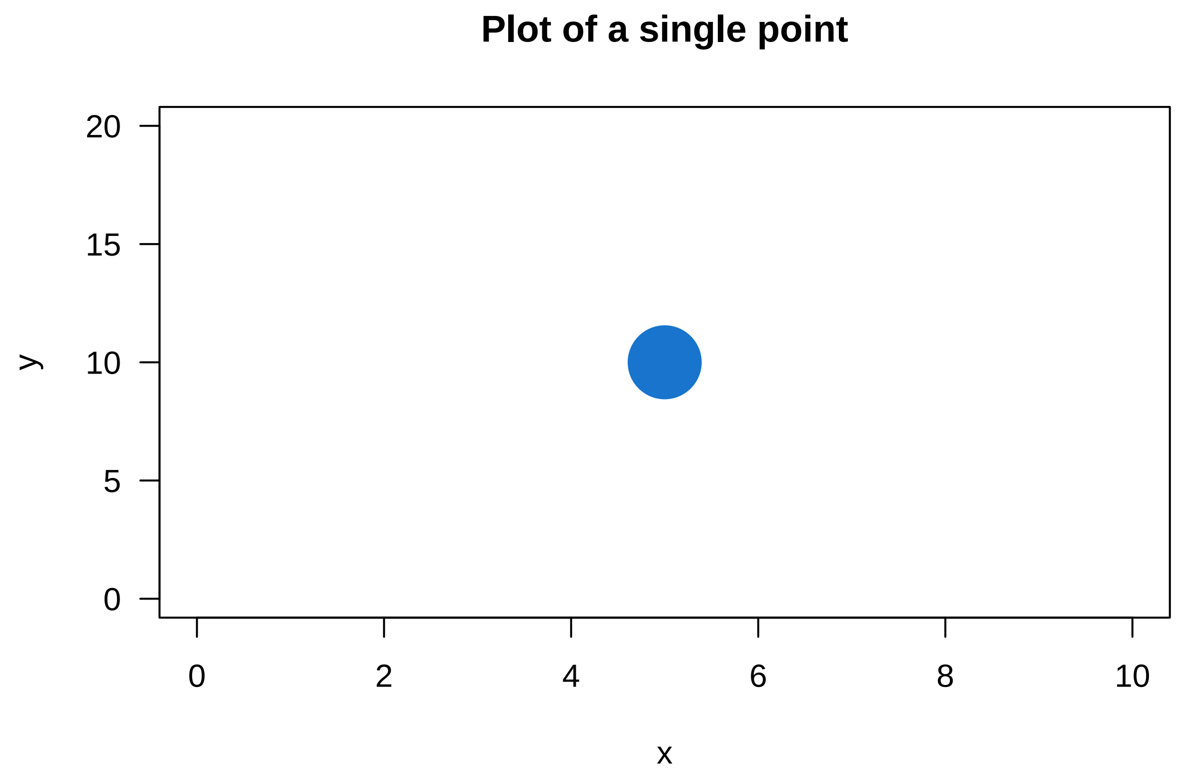
In my opinion, many of these shapes are not very useful.

It's a little bit like getting a cable TV bundle that has 500 channels, but you only watch 6 or 7 of them.

However, I encourage you to experiment and decide for yourself.

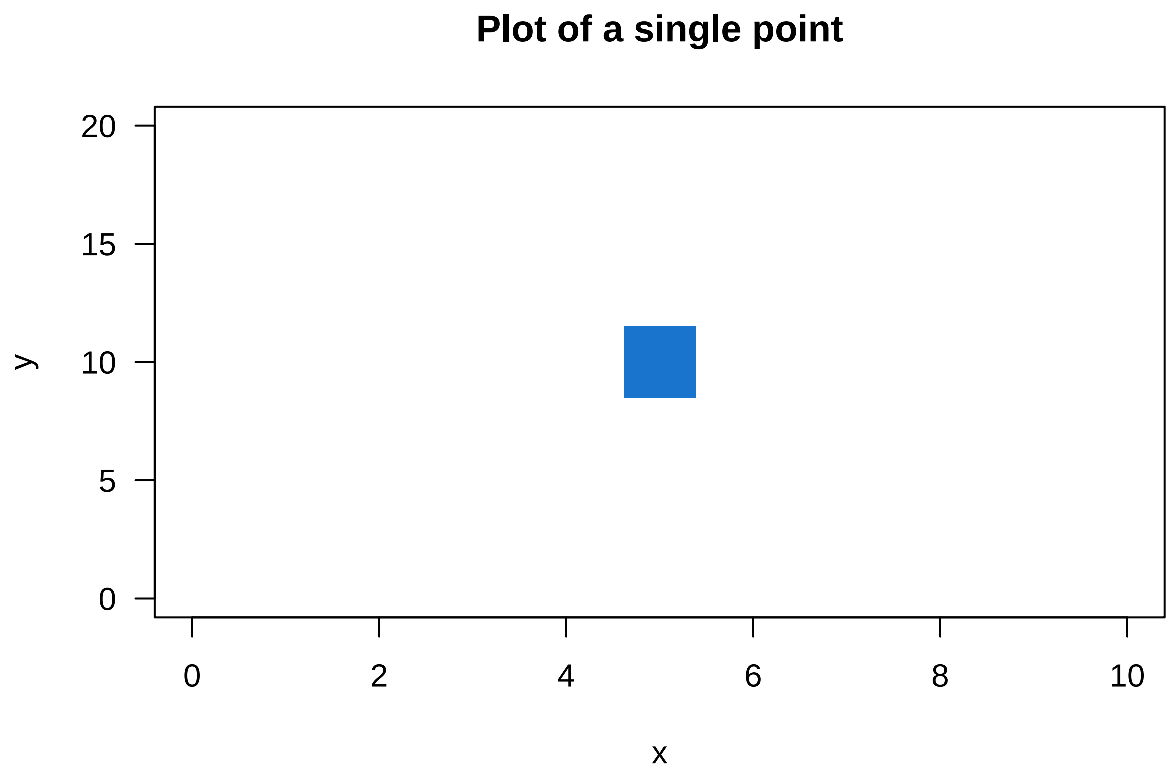
Personally, I think that the best shape for general plotting purposes is a solid circular disk, and you can obtain this by setting `pch` to the value 19:

```
plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 5,
    col = "dodgerblue3",
    pch = 19        # Here is where we specify
                    # a solid circular disk shape
                    # for the point
)
```



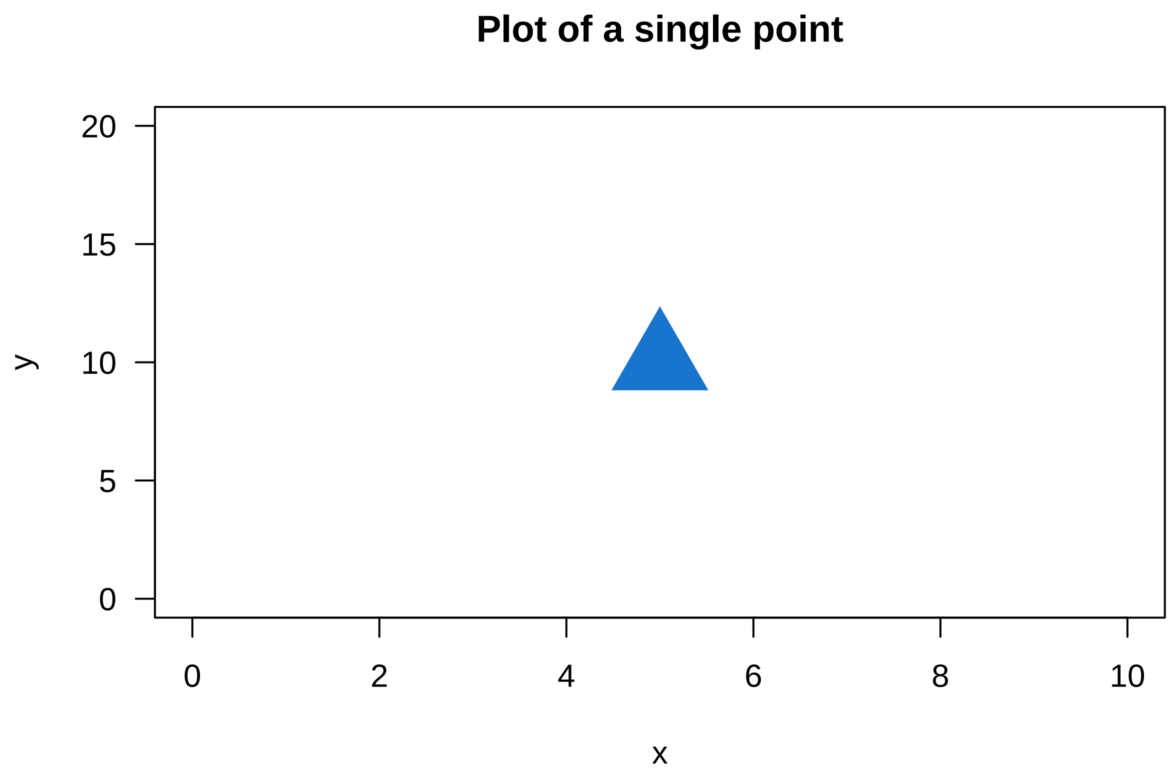
You can obtain a solid square shape by setting `pch` equal to 15:

```
plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 5,
    col = "dodgerblue3",
    pch = 15              # Here is where we specify
                          # a solid square shape
                          # for the point
)
```



You can obtain a solid triangle shape by setting `pch` equal to 17:

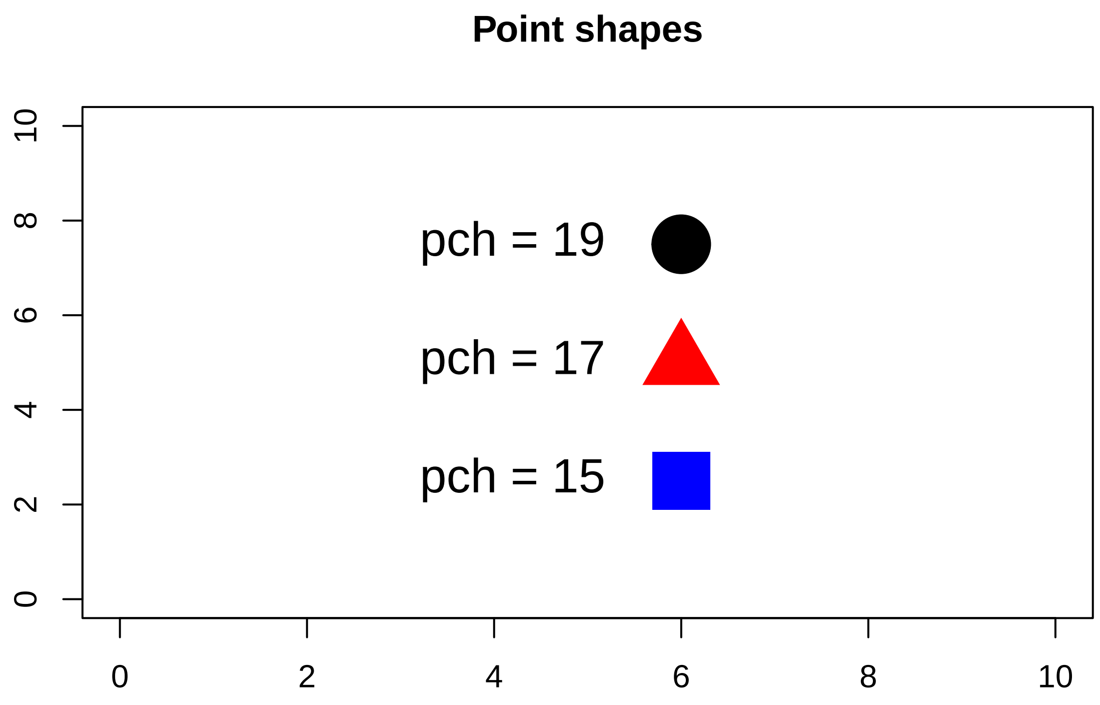
```
plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 5,
    col = "dodgerblue3",
    pch = 17              # Here is where we specify
                          # a solid triangular shape
                          # for the point
)
```



Let’s summarize these shapes:

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 10),
    main = "Point shapes",
    xlab = "",
    ylab = ""
)

text(
    x = 3,
    y = 7.5,
    labels = "pch = 19",
    pos = 4,
    cex = 1.5
)

points(
    x = 6,
    y = 7.5,
    pch = 19,
    cex = 4,
    col = "black"
)
```

```r
text(
    x = 3,
    y = 5,
    labels = "pch = 17",
    pos = 4,
    cex = 1.5
)

points(
    x = 6,
    y = 5,
    pch = 17,
    cex = 4,
    col = "red"
)



text(
    x = 3,
    y = 2.5,
    labels = "pch = 15",
    pos = 4,
    cex = 1.5
)

points(
    x = 6,
    y = 2.5,
    pch = 15,
    cex = 4,
    col = "blue"
)
```



There's something special about the shapes numbered from 21 to 25.

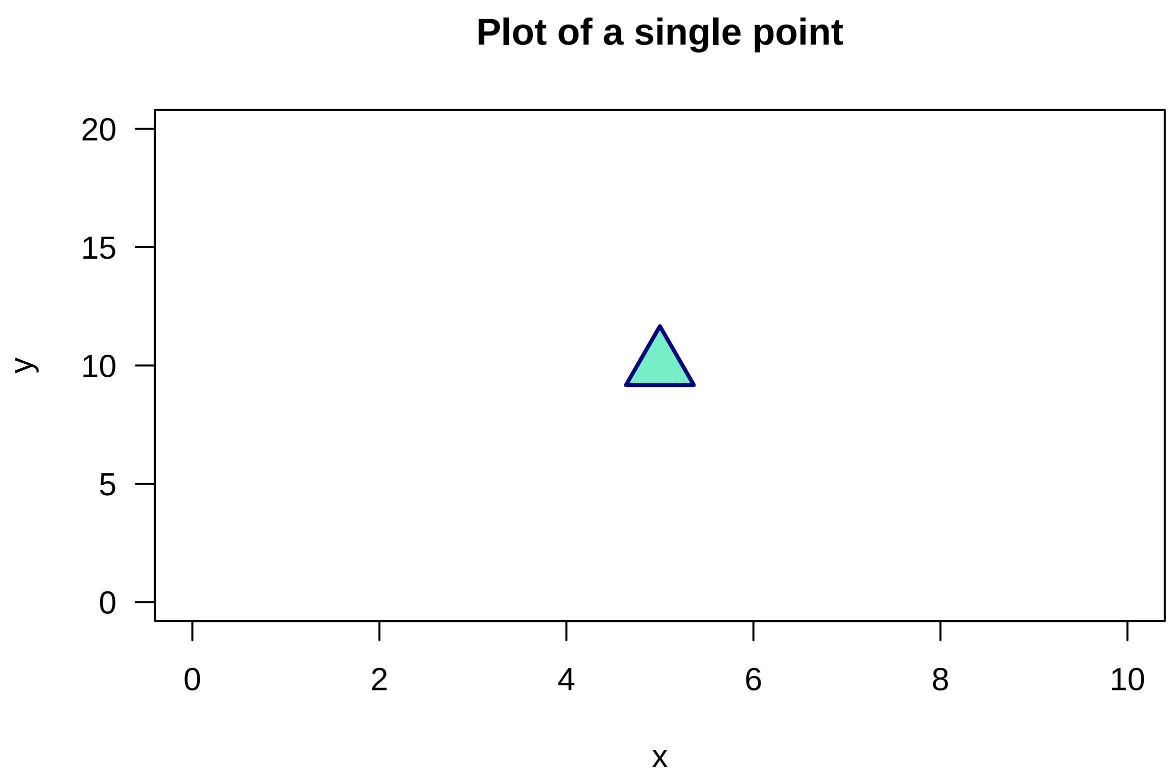
These shapes are comprised of an outline and a fill.

For these shapes, the `col =` option controls the color of the *outline*, and the `bg =` option controls the color of the fill.

The `lwd` option determines the width of the outline.

For instance, to draw a point at $x = 5$ and $y = 10$ using a triangle with an outline color of "navy", an outline width of 2, and a fill color of "aquamarine2", we have:

```
plot(
    x = 5,
    y = 10,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1,
    cex = 3.5,
    pch = 24,
    lwd = 2,
    col = "navy",           # Here is where we specify
                            # the outline color
    bg = "aquamarine2"      # Here is where we specify
                            # the fill color
)
```



So that's how we can adjust the shape of a point by using the `pch` option.

Now let's see how to use vectors to plot multiple points with a single function call.

Exercise 3.5: Adjusting the shape of a point

Create a graph of the single point with a solid square shape at $x = 7$ and $y = 12$, using a character expansion factor of 3.5 and the color "darkorchid3".

Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

```
# Type your solution in here
```

Exercise 3.6: Adjusting the shape of a point

Create a graph of a single circular point at $x = 7$ and $y = 12$, and use a character expansion factor of 3.5, an outline color of "darkred", an outline width of 2, and a fill color of "salmon1".

Use a `pch` value of 21 to obtain a circular point with a color fill.

Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

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

Section 6: Multiple points

Main Idea: *We can plot multiple points by using a single function call*

In this section, we'll learn how to plot multiple points by using a single function call.

So far, we've only created plots by plotting each point individually, so we had to use a separate function call for each point.

We can plot multiple points all at once by using a vector to hold the x values and another vector to hold the y values.

For instance, suppose we want to plot these 5 points:

Point	X	Y
1	2	8
2	3	4
3	5	7
4	9	11
5	12	13

We can think of the column of x values as a vector:

$$X = \langle 2, 3, 5, 9, 12 \rangle$$

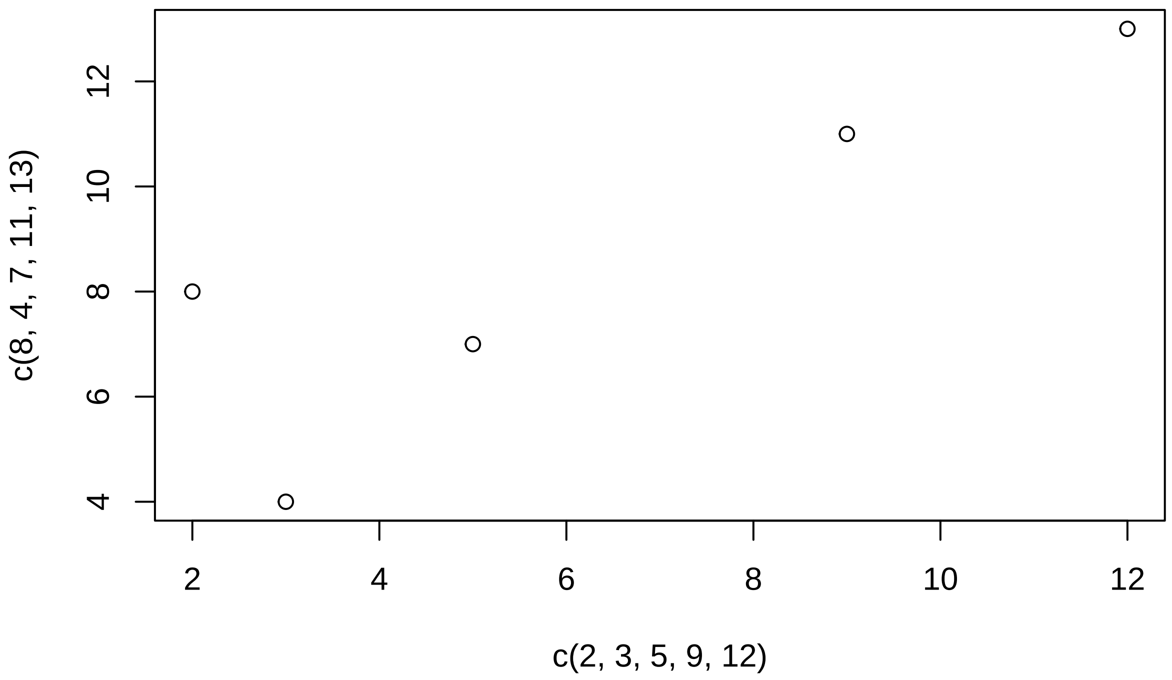
Likewise, we can think of the column of y values as a vector:

$$Y = \langle 8, 4, 7, 11, 13 \rangle$$

We can plot all of these points at once by using these two vectors.

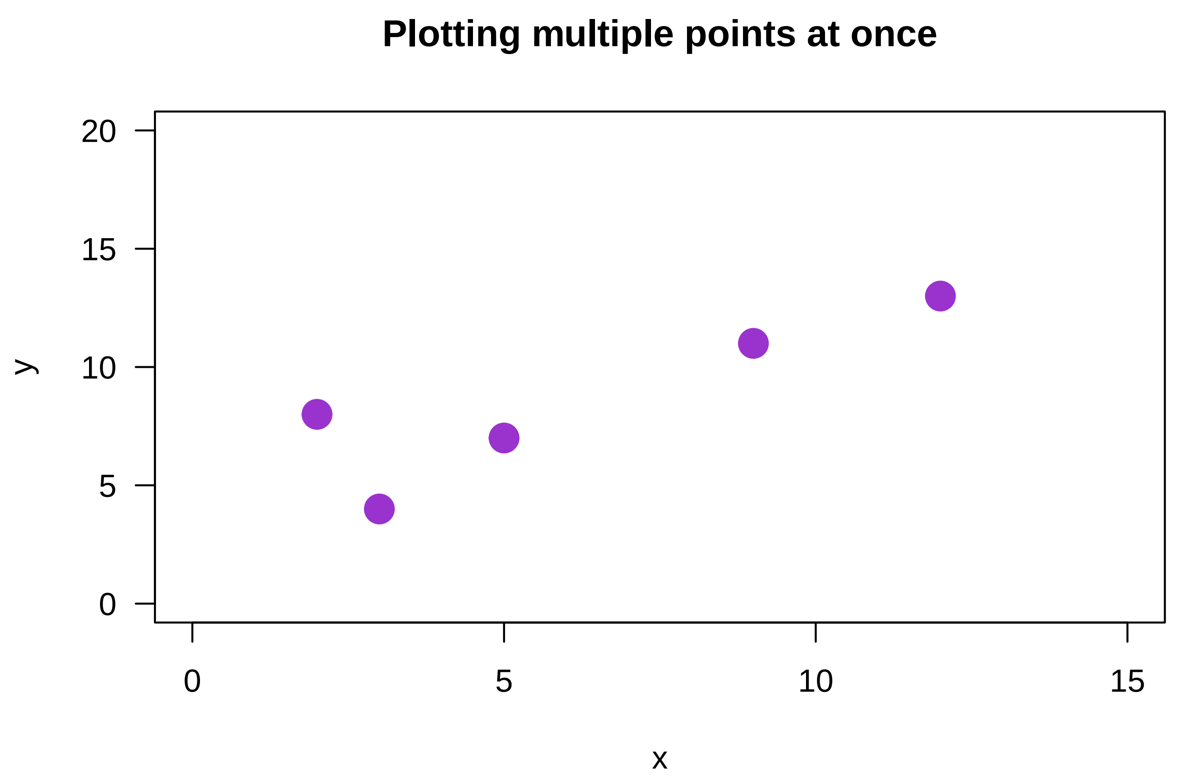
Let's first look at the most basic form of this graph:

```r
plot(
    x = c(2, 3, 5, 9, 12),
    y = c(8, 4, 7, 11, 13)
)
```



Now let's make this better:

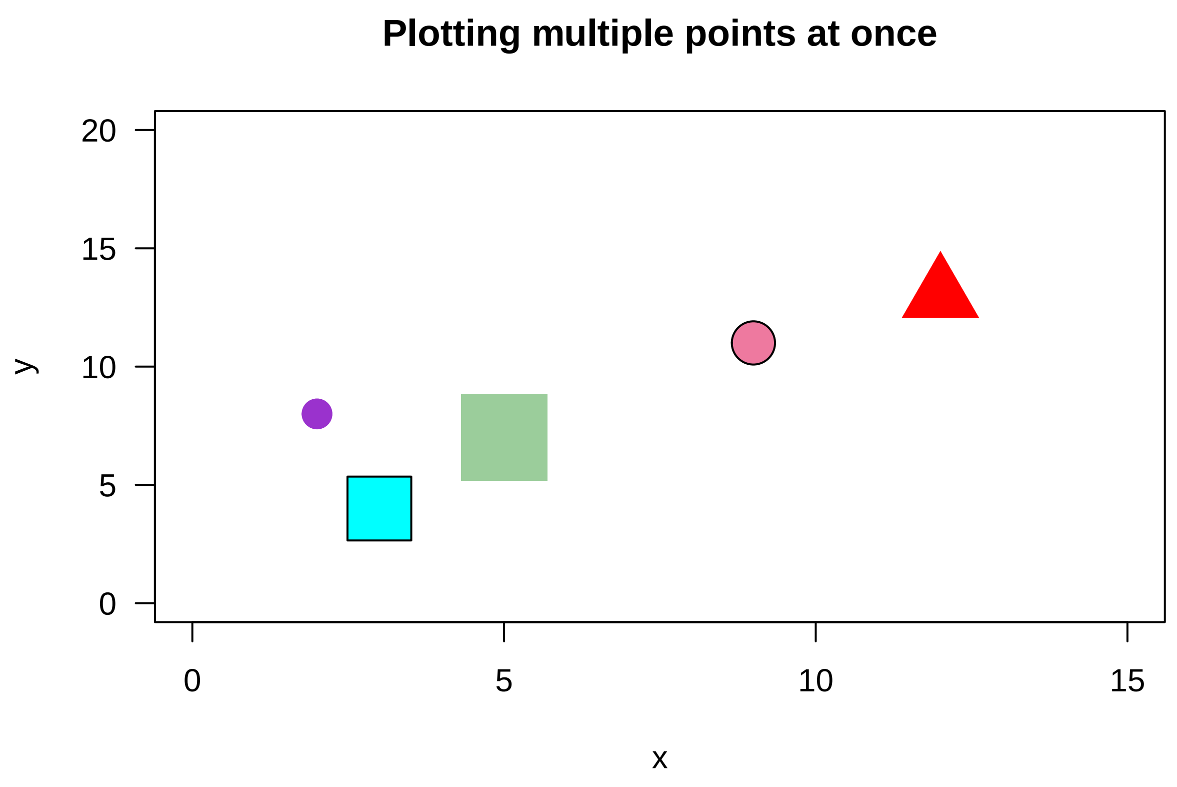
```r
plot(
    x = c(2, 3, 5, 9, 12),
    y = c(8, 4, 7, 11, 13),
    main = "Plotting multiple points at once",
    xlim = c(0, 15),
    ylim = c(0, 20),
    xlab = "x",
    ylab = "y",
    pch = 19,
    cex = 2,
    col = "darkorchid3",
    las = 1
)
```



Notice in this example that all the points have the same value for the point shape, size, and color.

We can even use vectors to specify each option for each point:

```
plot(
    x = c(2, 3, 5, 9, 12),
    y = c(8, 4, 7, 11, 13),
    main = "Plotting multiple points at once",
    xlim = c(0, 15),
    ylim = c(0, 20),
    xlab = "x",
    ylab = "y",
    pch = c(19, 22, 15, 21, 17),
    cex = c(2, 5, 6, 3, 4),
    col =
        c(
            "darkorchid3",
            "black",
            "darkseagreen3",
            "black",
            "red"
        ),
    bg =
        c(
            "black",
            "cyan1",
            "black",
            "palevioletred2",
            "black"
        ),
    las = 1
)
```



So that's how we can use vectors to plot multiple points using a single function call.

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

Exercise 3.7: Plotting multiple points using vectors

Using one call to the `plot()` function, plot these points:

X	*Y*
3	25
6	22
8	21
10	17
13	15

Solution

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

Exercise 3.8: Express your soul with modern art

Express the poetic yearnings of your soul by creating modern art:

- Create a completely blank plot, with x ranging from 0 to 10, and y ranging from 0 to 10 as well. Make the plot square.
- In the upper left quadrant, draw an upright triangle colored red. Make it big!
- In the upper right quadrant, draw an inverted triangle with a black border, and a yellow fill. Make it the same size as the first triangle.
- In the lower left quadrant, draw a circle with a black border and a green fill. Make it the same size as the other objects.
- Finally, in the lower right quadrant draw a solid square colored blue.

Solution

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

Module Review

In this module, we learned how to work with points.

- In Section 1, we saw how to add points to a pre-existing graphics plot.
- In Section 2, we found out how to annotate a plot with text.
- In Section 3, we learned how to adjust the size of a point.
- In Section 4, we saw how to modify the color of a point.
- In Section 5, we learned how to modify the shape of a point.
- In Section 6, we learned how to plot multiple points by using a single function call.

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

- Add points to a pre-existing plotting region by using the `points()` function.
- Annotate a graph with text by using the `text()` function.
- Adjust the size of a point by using the `cex` option.
- Modify the color of a point by using the `col` option.
- Control the shape of a point by using the `pch` option.
- Plot multiple points using a single call of the `plot()` function.

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

- `points()`
- `text()`

All right! That's it for Module 3: Points.

Now let's move on to Module 4: Lines and Polygons

Solutions to the Exercises

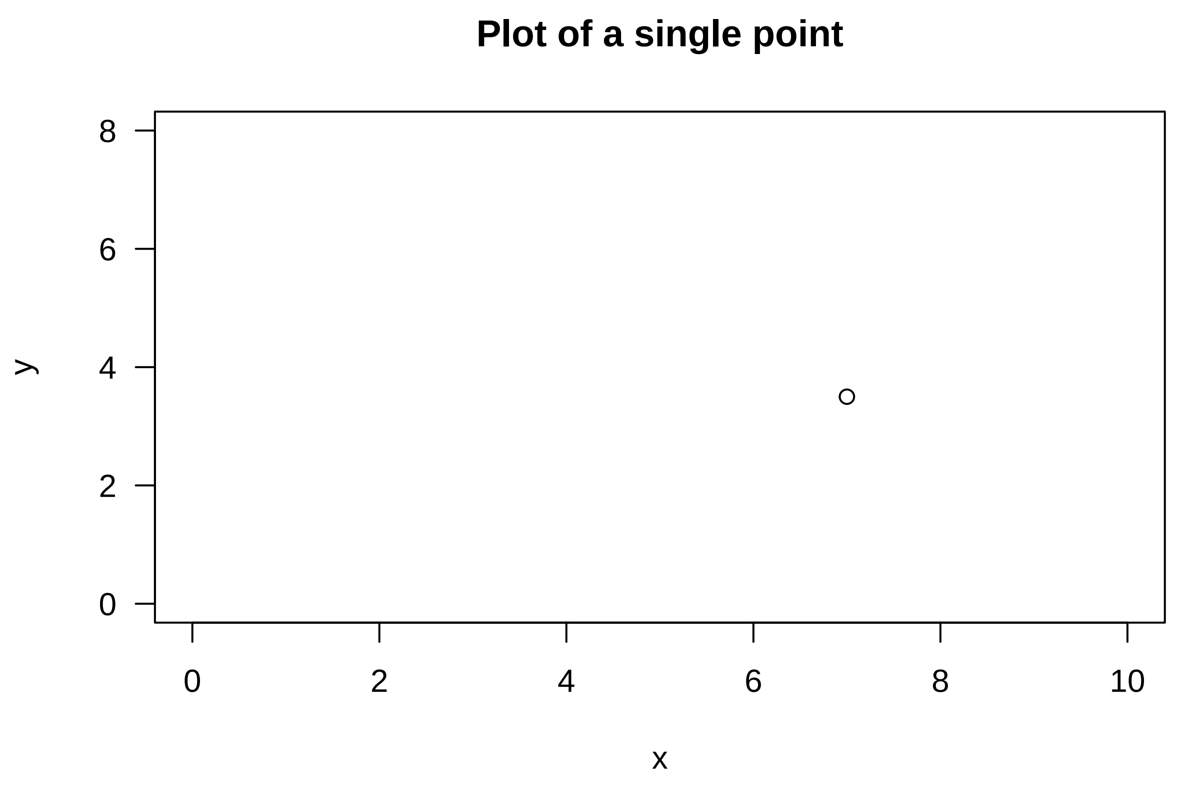
Exercise 3.1: Graphing a single point

First, create an empty plotting region with no data, where the x-values range from 0 to 10, and the y-values range from 0 to 8$.

Then use the `points()` function to draw a single point located at $x = 7$ and $y = 3$.

Solution

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 8),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1
)

points(
    x = 7,
    y = 3.5
)
```



Exercise 3.2: Annotating with text

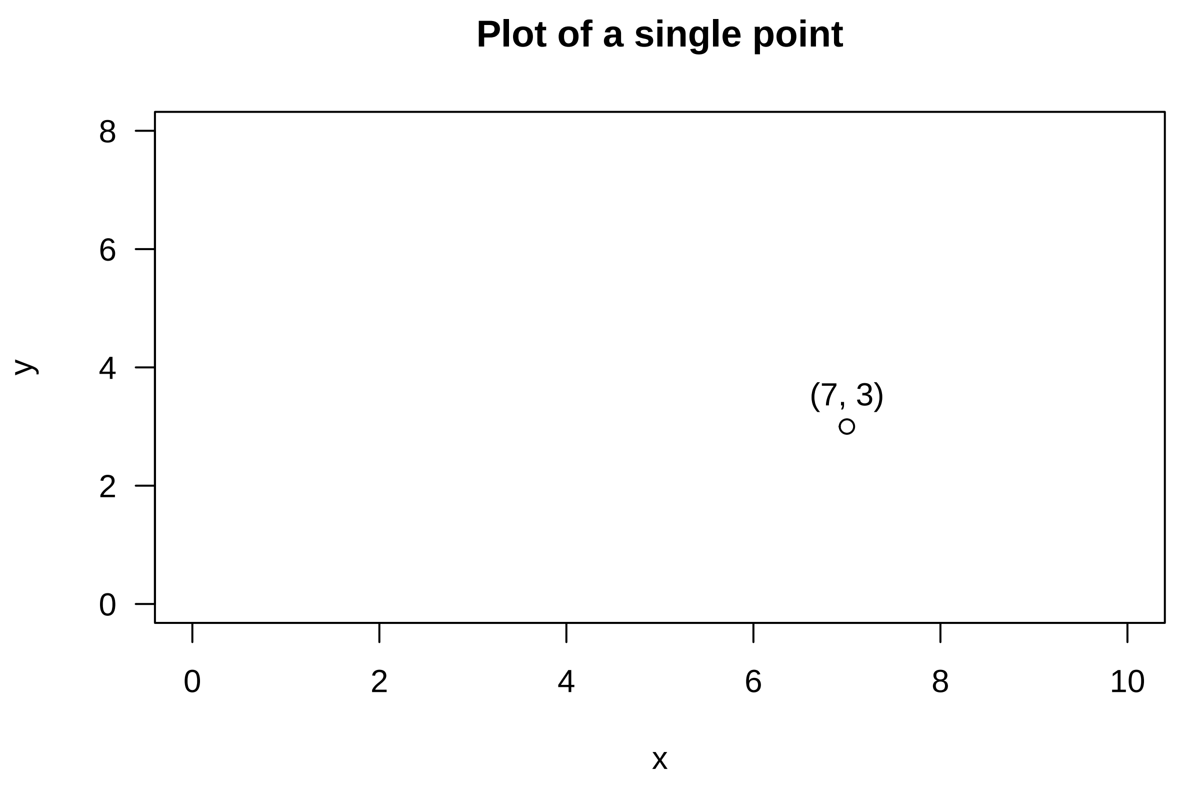
Draw a graph with a single point located at $x = 7$ and $y = 3$.

Set the x-values for the plotting region to range from 0 to 10, and the y-values to range from 0 to 8$.

Then annotate this point with its coordinates using the `text()` function.

Solution

```
plot(
    x = 7,
    y = 3,
    xlim = c(0, 10),
    ylim = c(0, 8),
    main = "Plot of a single point",
    xlab = "x",
    ylab = "y",
    las = 1
)

text(
    x = 7,
    y = 3.5,
    labels = "(7, 3)"
)
```



Exercise 3.3: Adjusting the size of a point

Create a graph of the single point at $x = 7$ and $y = 12$, and use a character expansion factor of 3.5.

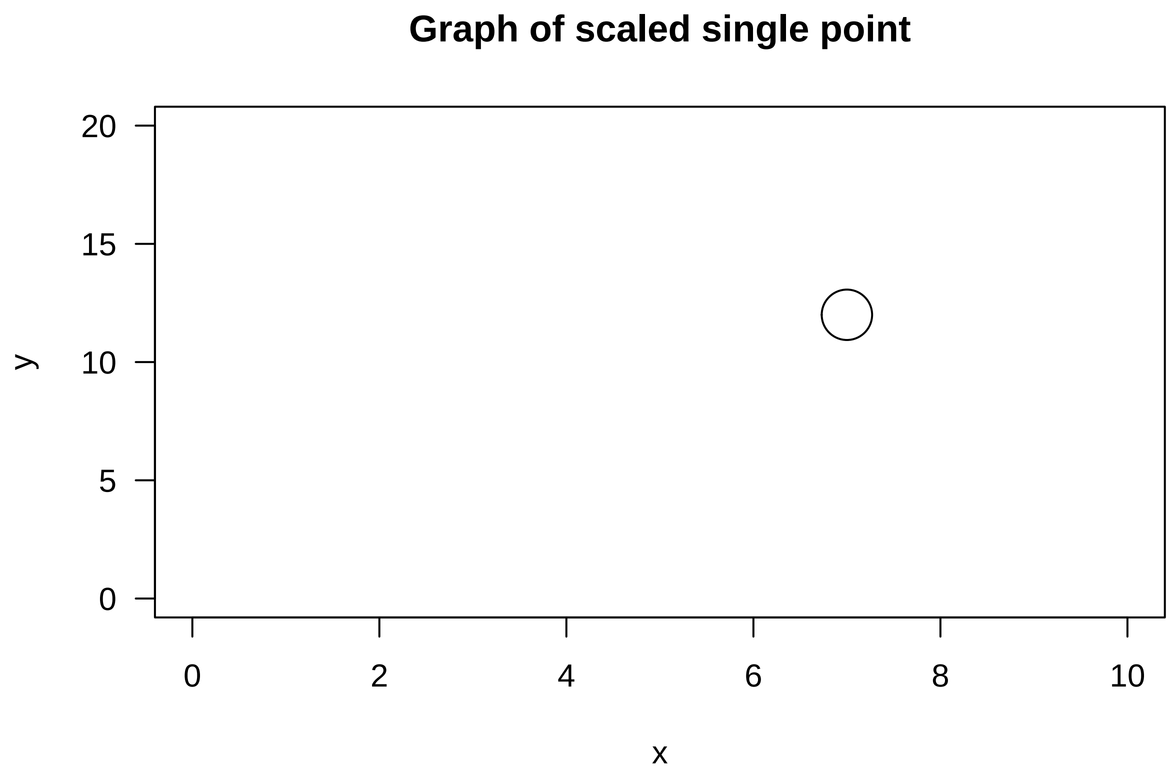
Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

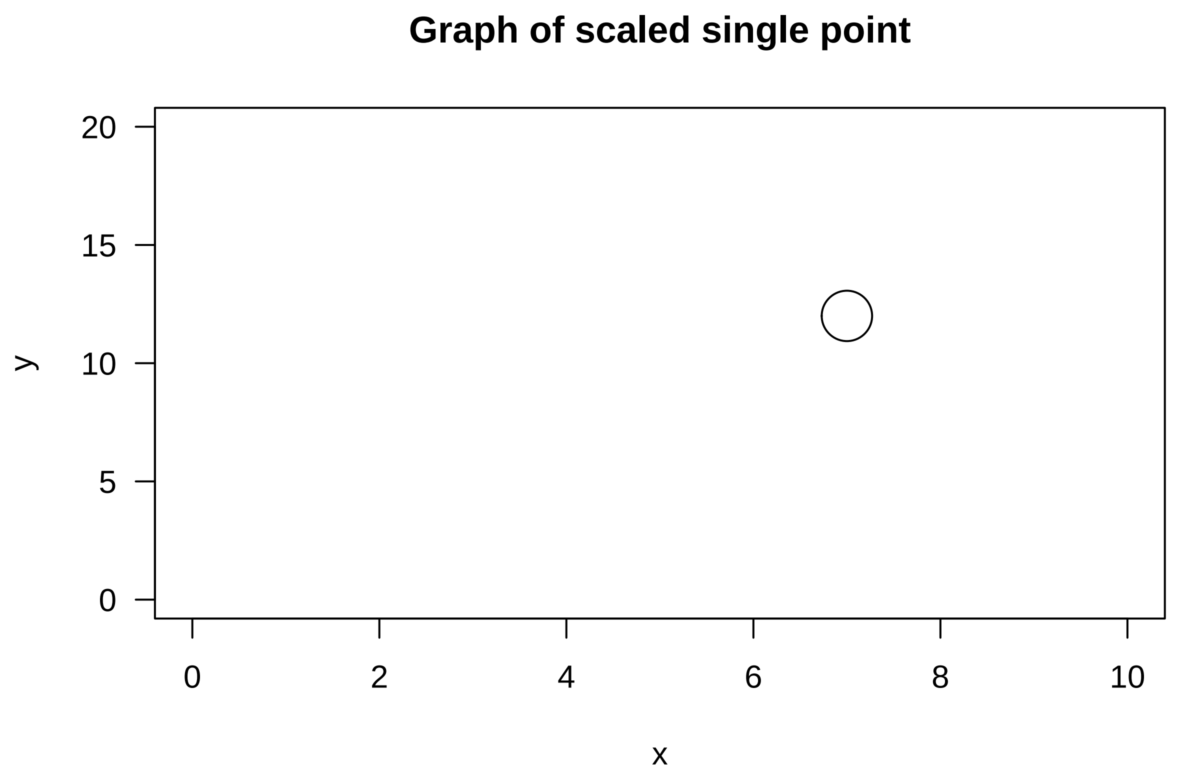
Here's one approach:

```r
plot(
    x = 7,
    y = 12,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Graph of scaled single point",
    xlab = "x",
    ylab = "y",
    cex = 3.5,
    las = 1
)
```



Here's another approach:

```r
# First, create an empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Graph of scaled single point",
    xlab = "x",
    ylab = "y",
    las = 1
)

# Then plot the point:

points(
    x = 7,
    y = 12,
    cex = 3.5
)
```



Exercise 3.4: Adjusting the color of a point

Create a graph of the single point at $x = 7$ and $y = 12$, using a character expansion factor of 3.5 and a color of "hotpink3".

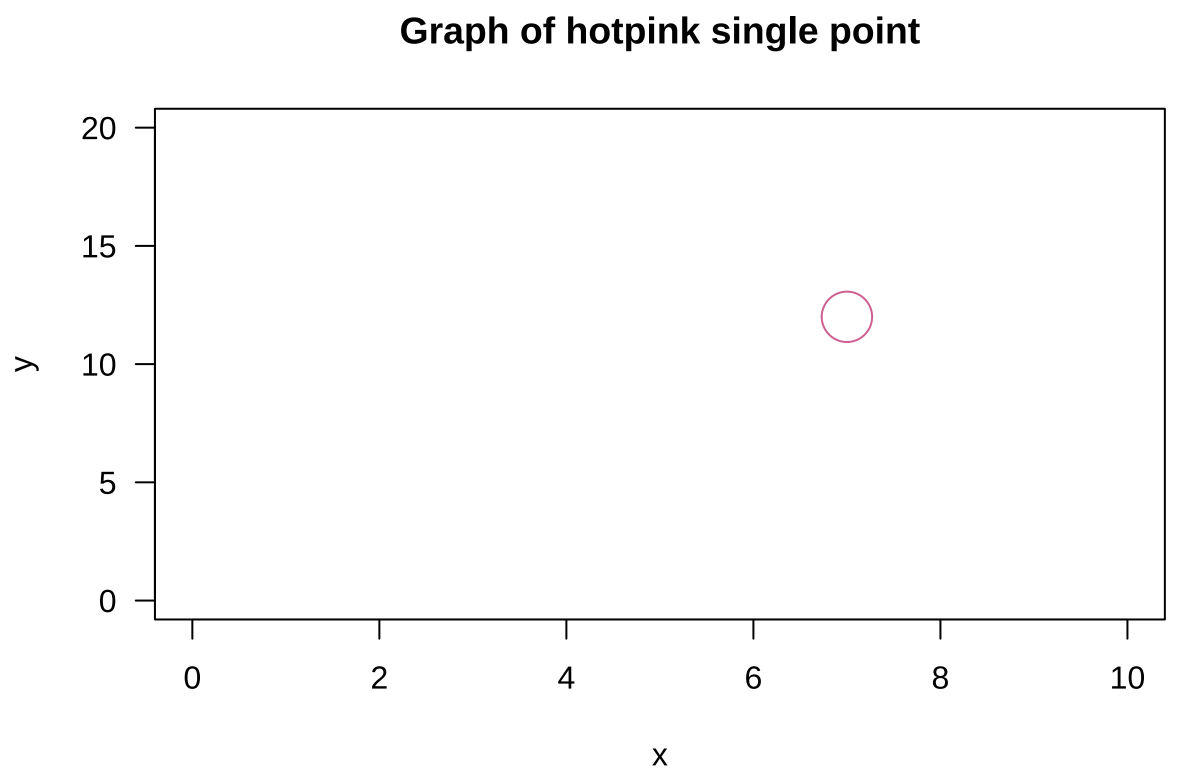
Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

Here's my solution:

```r
plot(
    x = 7,
    y = 12,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Graph of hotpink single point",
    xlab = "x",
    ylab = "y",
    cex = 3.5,
    col = "hotpink3",
    las = 1
)
```



Exercise 3.5: Adjusting the shape of a point

Create a graph of the single point with a solid square shape at $x = 7$ and $y = 12$, using a character expansion factor of 3.5 and the color "darkorchid3".

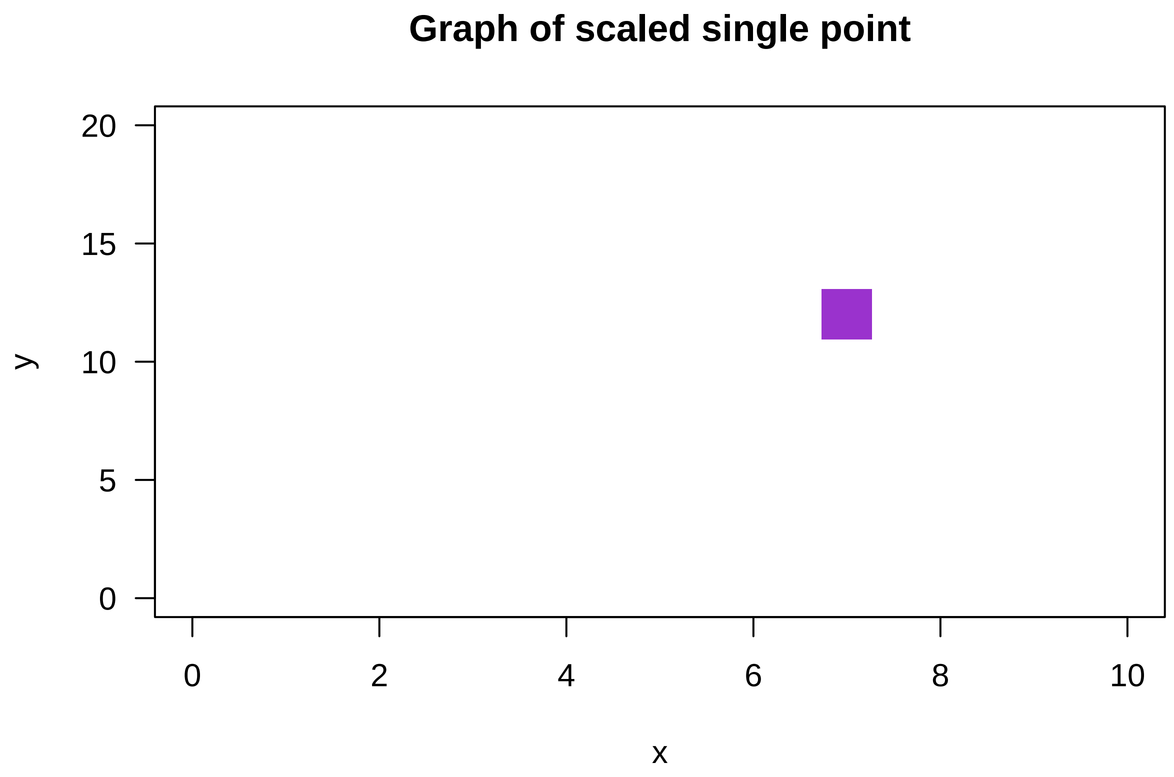
Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

Here's my solution:

```
plot(
    x = 7,
    y = 12,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Graph of scaled single point",
    xlab = "x",
    ylab = "y",
    cex = 3.5,
    col = "darkorchid3",
    pch = 15,
    las = 1
)
```



Exercise 3.6: Adjusting the shape of a point

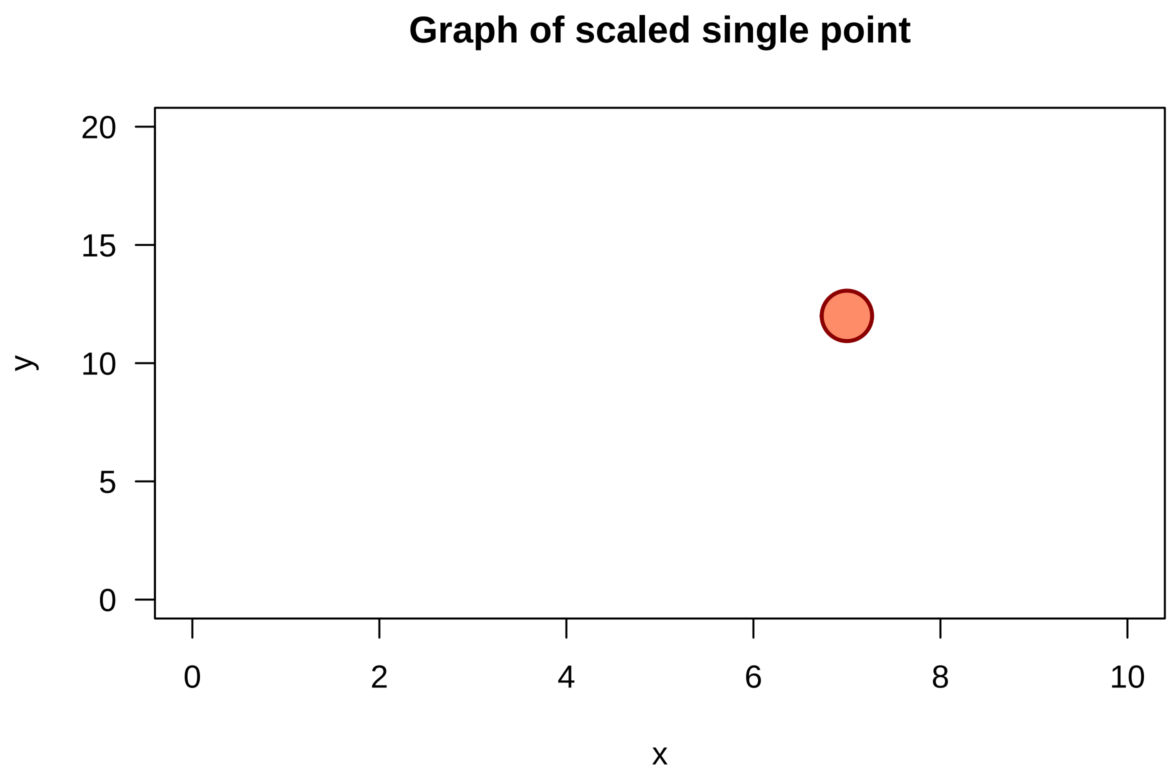
Create a graph of a single circular point at $x = 7$ and $y = 12$, and use a character expansion factor of 3.5, an outline color of "darkred", an outline width of 2, and a fill color of "salmon1".

Let the x-axis range from 0 to 10, and let the y-axis range from 0 to 20.

Remember to include a main title and to properly label the axes.

Solution

```
plot(
    x = 7,
    y = 12,
    xlim = c(0, 10),
    ylim = c(0, 20),
    main = "Graph of scaled single point",
    xlab = "x",
    ylab = "y",
    cex = 3.5,
    lwd = 2,
    col = "darkred",
    bg = "salmon1",
    pch = 21,
    las = 1
)
```



Exercise 3.7: Plotting multiple points using vectors

Using one call to the `plot()` function, plot these points:

X	Y
3	25
6	22
8	21
10	17
13	15

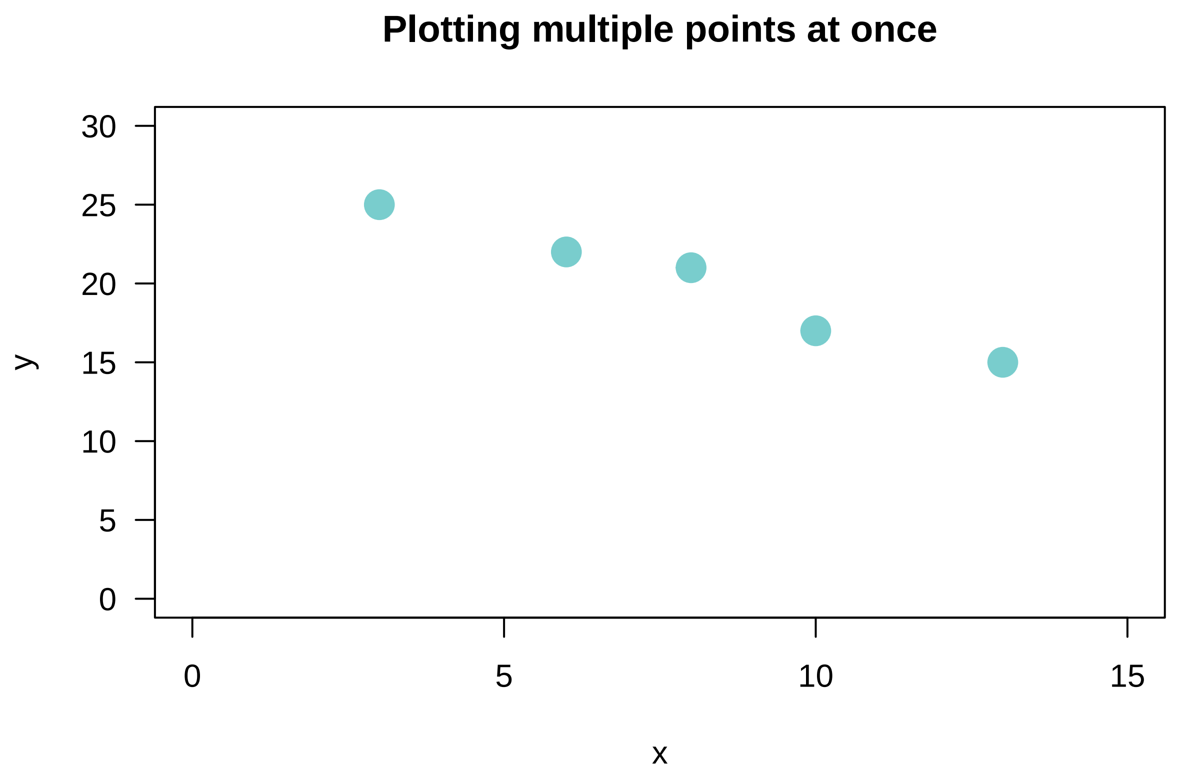
Solution

First, let's create the X and Y vectors:

```r
x.vector <- c(3, 6, 8, 10, 13)

y.vector <- c(25, 22, 21, 17, 15)
```

Now we can plot all these values in just one call to the `plot()` function:

```r
plot(
    x.vector,
    y.vector,
    main = "Plotting multiple points at once",
    xlim = c(0, 15),
    ylim = c(0, 30),
    xlab = "x",
    ylab = "y",
    pch = 19,
    cex = 2,
    col = "darkslategray3",
    las = 1
)
```



Exercise 3.8: Express your sould with modern art

Express the poetic yearnings of your soul by creating modern art:

- Create a completely blank plot, with x ranging from 0 to 10, and y ranging from 0 to 10 as well. Make the plot square.
- In the upper left quadrant, draw an upright triangle colored red. Make it big!
- In the upper right quadrant, draw an inverted triangle with a black border, and a yellow fill. Make it the same size as the same triangle.
- In the lower left quadrant, draw a circle with a black border and a green fill. Make it the same size as the other objects.
- Finally, in the lower right quadrant draw a solid square colored blue.

Solution

```
# First, to make the image a square,
# we set the graphical parameter `pty`
# by assigning the value "s" to it:

par( pty = "s" )
```

```r
# Next, create a completely blank plot

plot( x = NULL,
      xlim = c(0, 10),
      ylim = c(0, 10),
      axes = FALSE,
      xlab = "",
      ylab = ""
)


# Finally, draw the shapes:

points( 3, 7, pch = 17, cex = 10, col = "red" )
points( 7, 8, pch = 25, cex = 10, bg = "yellow", lwd = 2 )
points( 3, 3, pch = 21, cex = 10, bg = "green", lwd = 2 )
points( 7, 3, pch = 15, cex = 10, col = "blue" )
```