The Lemniscate Challenge Problem SOLUTIONS

First, let's clear the global computing environment:

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

Problem 7: The Lemniscate of Bernoulli

I love this problem, because it illustrates a beautiful application of the ideas of vectorized operations and the `seq()` function. You might not be interested in the actual content of the problem itself, but I hope you will learn some useful techniques for your own practice.

A "lemniscate" is a curve that is shaped like a figure 8 on its side or an infinity symbol.

There are many different kinds of lemniscates, and a great reference for these different curves is the Wikipedia entry for "lemniscate".

We are going to draw the so-called "Lemniscate of Bernoulli".

There are many ways to specify this lemniscate, but we will focus on what are called *parameteric equations*.

That means that we have one numeric variable, which is the "parameter", and then we calculate the values of x and y based on this parameter.

By varying the value of the parameter, we obtain a set of paired values of x and y.

These paired values are coordinated data, because the x and y values correspond to the same value of the parameter.

To create a lemniscate, we use the parameter θ with a range from 0 to 2π.

The formula for x is:

$$x(\theta) = \frac{\cos(\theta)}{1 + \sin^2(\theta)}$$

The formula for y is:

$$x(\theta) = \frac{\sin(\theta) \cdot \cos(\theta)}{1 + \sin^2(\theta)}$$

In this problem, we'll first get familiar with these parametric equations by calculating a few points by hand and graphing them. Then we'll see how to use the `seq()` function and vectorized operations to automate this process.

Part (a): First point

Using the formulas for $x(\theta)$ and $y(\theta)$, calculate the value of x and y for $\theta = 0$. Store these two values in separate variables. Then report each result using a `cat()` statement, displaying the value with 3 decimal places.

Hint: store the value of the parameter in a variable, and then use this variable throughout your code. Then you can do the next two parts easily by just copying and pasting the code block and making a few modifications.

Solution

First, let's define the parameter value:

```
parameter.value <- 0
```

Now we can calculate the value of the x-coordinate:

```
point.1.x <-
  cos( parameter.value ) / (1 + (sin(parameter.value))^2)

cat(
  "x-coordinate of first point:",
  formatC(
    x = point.1.x,
    format = "f",
    digits = 3
  )
)
```

```
## x-coordinate of first point: 1.000
```

Now we can calculate the value of the y-coordinate:

```
parameter.value <- 0

point.1.y <-
  cos( parameter.value ) * sin( parameter.value ) /
  (1 + (sin(parameter.value))^2)

cat(
  "y-coordinate of first point:",
  formatC(
    x = point.1.y,
    format = "f",
    digits = 3
  )
)
```

```
## y-coordinate of first point: 0.000
```

Part (b): Second point

Using the formulas for $x(\theta)$ and $y(\theta)$, calculate the value of x and y for $\theta = \pi/6$. Store these two values in separate variables. Then report each result using a `cat()` statement, displaying the value with 3 decimal places.

Remember, if you designed your code well in part (a), you can just copy and paste the code blocks and then modify a few things to obtain the answers for this part.

Solution

First, let's define the parameter value:

```
parameter.value <- pi / 6
```

Now we can calculate the value of the x-coordinate:

```
point.2.x <-
  cos( parameter.value ) / (1 + (sin(parameter.value))^2)

cat(
  "x-coordinate of second point:",
  formatC(
    x = point.2.x,
    format = "f",
    digits = 3
  )
)
```

```
## x-coordinate of second point: 0.693
```

Now we can calculate the value of the y-coordinate:

```
point.2.y <-
  cos( parameter.value ) * sin( parameter.value ) /
  (1 + (sin(parameter.value))^2)

cat(
  "y-coordinate of second point:",
  formatC(
    x = point.2.y,
    format = "f",
    digits = 3
  )
)
```

```
## y-coordinate of second point: 0.346
```

Part (c): Third point

Using the formulas for $x(\theta)$ and $y(\theta)$, calculate the value of x and y for $\theta = \pi/3$. Store these two values in separate variables. Then report each result using a `cat()` statement, displaying the value with 3 decimal places.

Remember, if you designed your code well in part (a), you can just copy and paste the code blocks and then modify a few things to obtain the answers for this part.

Solution

First, let's define the parameter value:

```
parameter.value <- pi / 3
```

Now we can calculate the value of the x-coordinate:

```
point.3.x <-
  cos( parameter.value ) / (1 + (sin(parameter.value))^2)

cat(
  "x-coordinate of third point:",
  formatC(
    x = point.3.x,
    format = "f",
    digits = 3
  )
)
```

```
## x-coordinate of third point: 0.286
```

Now we can calculate the value of the y-coordinate:

```
point.3.y <-
  cos( parameter.value ) * sin( parameter.value ) /
  (1 + (sin(parameter.value))^2)

cat(
  "y-coordinate of third point:",
  formatC(
    x = point.3.y,
    format = "f",
    digits = 3
  )
)
```

```
## y-coordinate of third point: 0.247
```

Part (d): Create the x-vector

Take the three x-values you created in parts (a) through (c) and put them into a vector. Save this vector in a variable, and report it using a `cat()` statement with all values formatted with 3 decimal places.

Solution

```
x.vector <-
  c(
    point.1.x,
    point.2.x,
    point.3.x
  )

cat(
  "x.vector: ",
```

```
  formatC(
    x = x.vector,
    format = "f",
    digits = 3
  )
)
```

```
## x.vector:  1.000 0.693 0.286
```

Part (e): Create the y-vector

Take the three y-values you created in parts (a) through (c) and put them into a vector. Save this vector in a variable, and report it using a `cat()` statement with all values formatted with 3 decimal places.

Solution

```
y.vector <-
  c(
    point.1.y,
    point.2.y,
    point.3.y
  )

cat(
  "y.vector: ",
  formatC(
    x = y.vector,
    format = "f",
    digits = 3
  )
)
```

```
## y.vector:  0.000 0.346 0.247
```

Part (f): Draw a graph

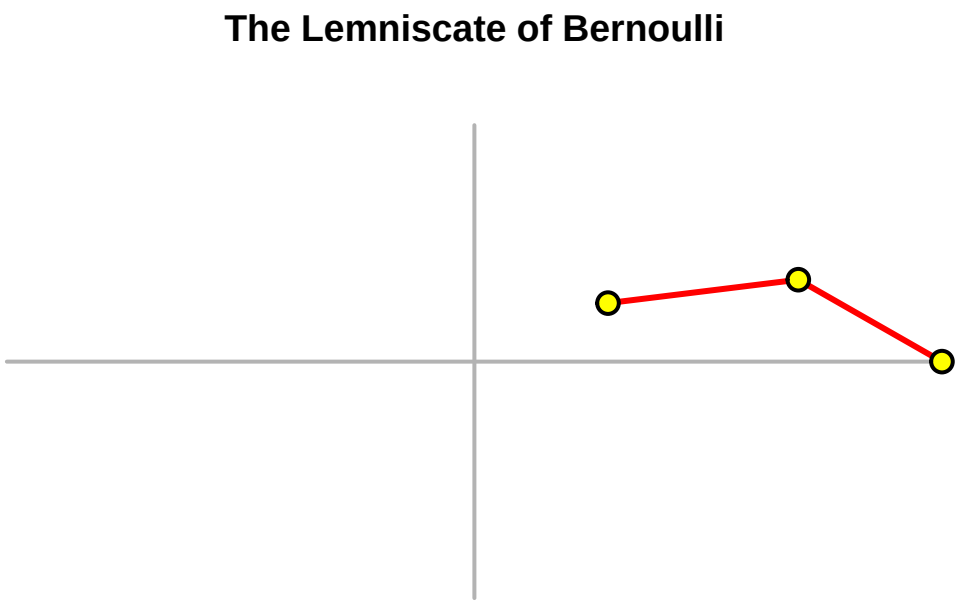
Now let's graph the line segments corresponding to these three points.

- Create a completely blank plotting display with no axes, surrounding box, or axis title, but include a main title such as "The Lemniscate of Bernoulli".
- The range of the x-axis should be from -1 to 1, and the range of the y-axis should be from -1 to 1.
- Include horizontal and vertical reference lines.
- Then use the `lines()` function to plot the two line segments.
- Finally, draw in the three points.

Solution

```r
# First, we'll create the plotting region

plot(
  x = NULL,
  xlim = c(-1, 1),
  ylim = c(-1, 1),
  main = "The Lemniscate of Bernoulli",
  xlab = "",
  ylab = "",
  axes = FALSE
)

# Drawing in the horizontal reference line

segments(
  x0 = -1,
  y0 = 0,
  x1 = 1,
  y1 = 0,
  lty = "solid",
  lwd = 2,
  col = "gray70"
)

# Drawing in the vertical reference line

segments(
  x0 = 0,
  y0 = -1,
  x1 = 0,
  y1 = 1,
  lty = "solid",
  lwd = 2,
  col = "gray70"
)


# Drawing in the line segments

lines(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  col = "red"
)

# Drawing in the points

points(
  x = x.vector,
  y = y.vector,
  pch = 21,
```

```
  lwd = 2,
  cex = 1.5,
  bg = "yellow",
  col = "black"
)
```



Part (g): Constructing the parameter vector

At this point, you should understand the basic strategy: pick a range of parameter values, calculate the values of x and y for each particular value, and then graph the corresponding line segments using the `lines()` segment.

However, our graph doesn't look very much like those nice pictures in the article in Wikipedia.

That's because we only use three points.

But that was a lot of work! We had to write individual code chunks for each calculation, and then create dedicated variables for each x and y value.

Now we're going to see how to use the `seq()` function and vectorized operations to automate this process and allow us to easily generate many points with no extra work.

We'll start by creating 20 line segments for the lemniscate instead of just 2.

Remember that we needed 3 points to specify two connected line segments.

Thus, to specify 20 connected line segments, we'll need 21 points.

Finally, remember that the parameter θ ranges from 0 to 2π.

So, for this part, you have to create a numeric vector using the `seq()` function with 21 values ranging from 0 to 2π. Store this vector in a variable, and report the first 5 values using a `cat()` statement, formatting each value with 3 decimal places.

Solution

```
parameter.value.vector <-
  seq( from = 0, to = 2 * pi, length.out = 21 )

cat(
  "Parameter value vector:",
  formatC(
    x = head( parameter.value.vector, n = 5 ),
    format = "f",
    digits = 3
  )
)
```

```
## Parameter value vector: 0.000 0.314 0.628 0.942 1.257
```

Part (h): x-value vector

Using the vector of parameter values that you created in part (g), use vectorized operations to create a vector of corresponding x-values. (Hint: remember the parametric formula!) Store this vector in a variable, and report the first 5 values using a `cat()` statement, formatting each value with 3 decimal places.

Solution

```
x.vector <-
  cos( parameter.value.vector ) /
  (1 + (sin(parameter.value.vector)^2))

cat(
  "x vector:",
  formatC(
    x = head( x.vector, n = 5 ),
    format = "f",
    digits = 3
  )
)
```

```
## x vector: 1.000 0.868 0.601 0.355 0.162
```

Part (i): y-value vector

Using the vector of parameter values that you created in part (g), use vectorized operations to create a vector of corresponding y-values. (Hint: remember the parametric formula!) Store this vector in a variable, and report the first 5 values using a `cat()` statement, formatting each value with 3 decimal places.

Solution

```r
y.vector <-
  cos( parameter.value.vector ) * sin( parameter.value.vector ) /
  (1 + (sin(parameter.value.vector)^2))

cat(
  "y vector:",
  formatC(
    x = head( y.vector, n = 5 ),
    format = "f",
    digits = 3
  )
)
```

```
## y vector: 0.000 0.268 0.353 0.287 0.154
```

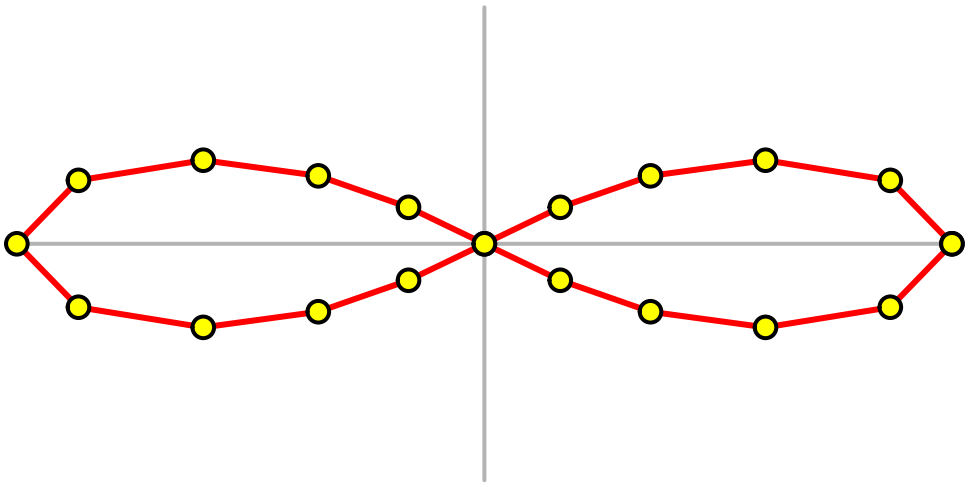
Part (j): Draw the graph

Copy your code from part (f) for drawing the graph of the lemniscate. Then run it using the x-vector and y-vector from parts (h) and (i), respectively.

Solution

```r
# First, we'll create the plotting region

plot(
  x = NULL,
  xlim = c(-1, 1),
  ylim = c(-1, 1),
  main = "The Lemniscate of Bernoulli",
  xlab = "",
  ylab = "",
  axes = FALSE
)

# Drawing in the horizontal reference line

segments(
  x0 = -1,
  y0 = 0,
  x1 = 1,
  y1 = 0,
  lty = "solid",
  lwd = 2,
  col = "gray70"
)

# Drawing in the vertical reference line

segments(
  x0 = 0,
  y0 = -1,
  x1 = 0,
  y1 = 1,
```

```
  lty = "solid",
  lwd = 2,
  col = "gray70"
)


# Drawing in the line segments

lines(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  col = "red"
)

# Drawing in the points

points(
  x = x.vector,
  y = y.vector,
  pch = 21,
  lwd = 2,
  cex = 1.5,
  bg = "yellow",
  col = "black"
)
```



Part (k): Drawing the graph

Almost done!

Our graph in part (j) looks better than what we achieved in part (f), because we were able to draw in more points.

But it's still not the beautiful smooth curve you see in the Wikipedia article.

That's because using only 20 line segments still doesn't provide enough resolution.

How about 1000 points? I bet that would work a lot better.

For this part, you should use the `seq()` function to construct a parameter value vector with 1001 points, and then use vectorized operations to construct x and y vectors with 1001 points.

Finally, draw a graph with these high-resolution vectors. Just draw the line segment, and forget about drawing the individual points, because there are too many of them. This time it should look a lot better.

I'm going to leave this up to you to organize. However, if you wrote your code well in the previous parts, you should be able to do this problem by copying and pasting and then changing one value. Good luck!

Solution

```
parameter.value.vector <-
  seq( from = 0, to = 2 * pi, length.out = 1000 )

x.vector <-
  cos( parameter.value.vector ) /
```

```
  (1 + (sin(parameter.value.vector)^2))

y.vector <-
  cos( parameter.value.vector ) * sin( parameter.value.vector ) /
  (1 + (sin(parameter.value.vector)^2))
```

```
plot(
  x = NULL,
  xlim = c(-1, 1),
  ylim = c(-1, 1),
  main = "The Lemniscate of Bernoulli",
  xlab = "",
  ylab = "",
  axes = FALSE
)

segments(
  x0 = 0,
  y0 = -1,
  x1 = 0,
  y1 = 1,
  lty = "solid",
  lwd = 2,
  col = "gray60"
)


segments(
  x0 = -1,
  y0 = 0,
  x1 = 1,
  y1 = 0,
  lty = "solid",
  lwd = 2,
  col = "gray60"
)



lines(
  x.vector,
  y.vector,
  lty = "solid",
  lwd = 3,
  col = "red"
)
```

The Lemniscate of Bernoulli