Problem Set 1 Solutions

CSCI E-5a: Programming in R

Let's clear the global computing environment:

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

Problem 1: Triangles

In this problem, we'll draw the same triangle using three different techniques. The triangle will have vertices at three points denoted A, B, and C, with these coordinates:

Point	x	y
A	1	2
B	2	7
C	5	3

Part (a): Using line segments

In the first method, we'll draw a triangle by using the `segments()` function three times:

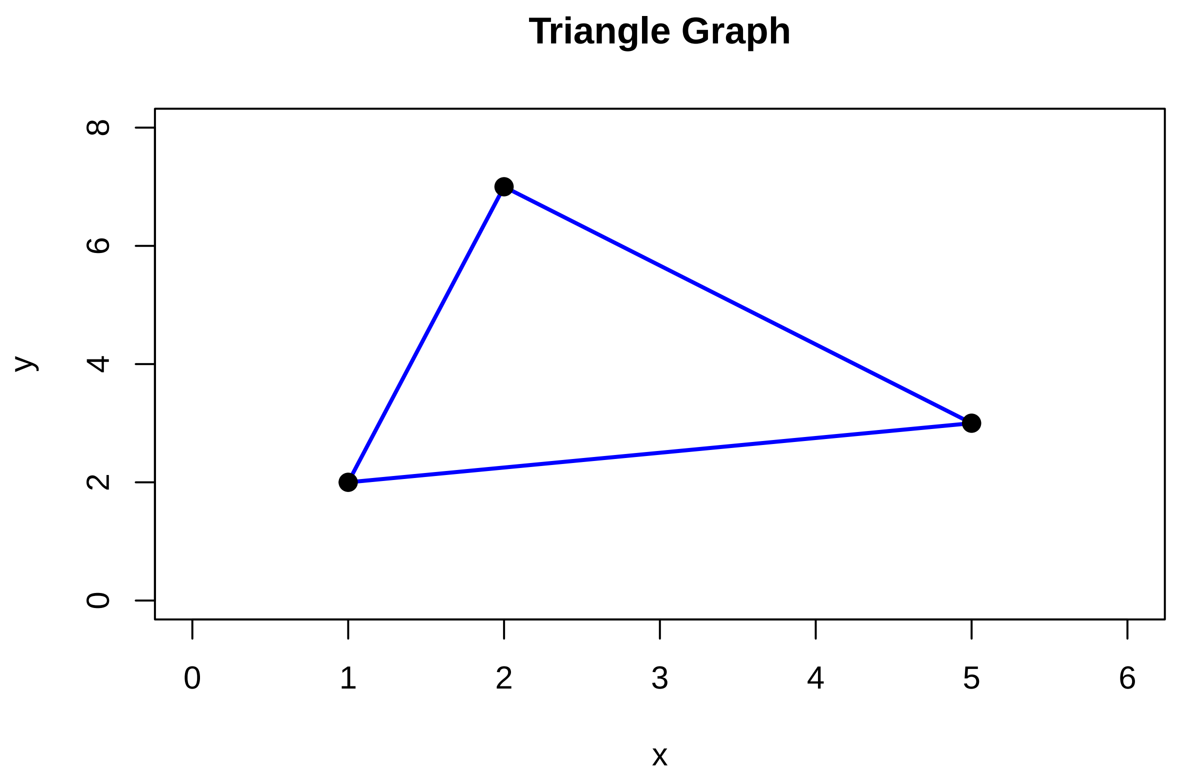
- First, create an empty plot with no data, where the x-axis ranges from 0 to 6 and the y-axis ranges from 0 to 8:
- Second, draw a line segment from the point $A = (1, 2)$ to the point $B = (2, 7)$.
- Next, draw a line segment from the point $B = (2, 7)$ to the point $C = (5, 3)$.
- Finally draw a line segment from the point $C = (5, 3)$ back to $A = (1, 2)$.

The lines of the triangle should be colored blue, with a line width of 2. Finally, draw a black circular point at each of the vertices of the triangle.

Solution

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

plot(
    x = NULL,
    xlim = c(0, 6),
    ylim = c(0, 8),
    main = "Triangle Graph",
    xlab = "x",
    ylab = "y"
)


# Now we can draw the line segments:

segments(
    x0 = 1,
    y0 = 2,
    x1 = 2,
    y1 = 7,
    lty = "solid",
    lwd = 2,
    col = "blue"
)

segments(
    x0 = 2,
    y0 = 7,
    x1 = 5,
    y1 = 3,
    lty = "solid",
    lwd = 2,
    col = "blue"
)

segments(
    x0 = 5,
    y0 = 3,
    x1 = 1,
    y1 = 2,
    lty = "solid",
    lwd = 2,
    col = "blue"
)


# Now we can draw the points:

points(
    x = 1,
    y = 2,
    pch = 19,
    cex = 1.2,
    col = "black"
)

points(
    x = 2,
    y = 7,
    pch = 19,
    cex = 1.2,
    col = "black"
)
```

```r
points(
    x = 5,
    y = 3,
    pch = 19,
    cex = 1.2,
    col = "black"
)
```

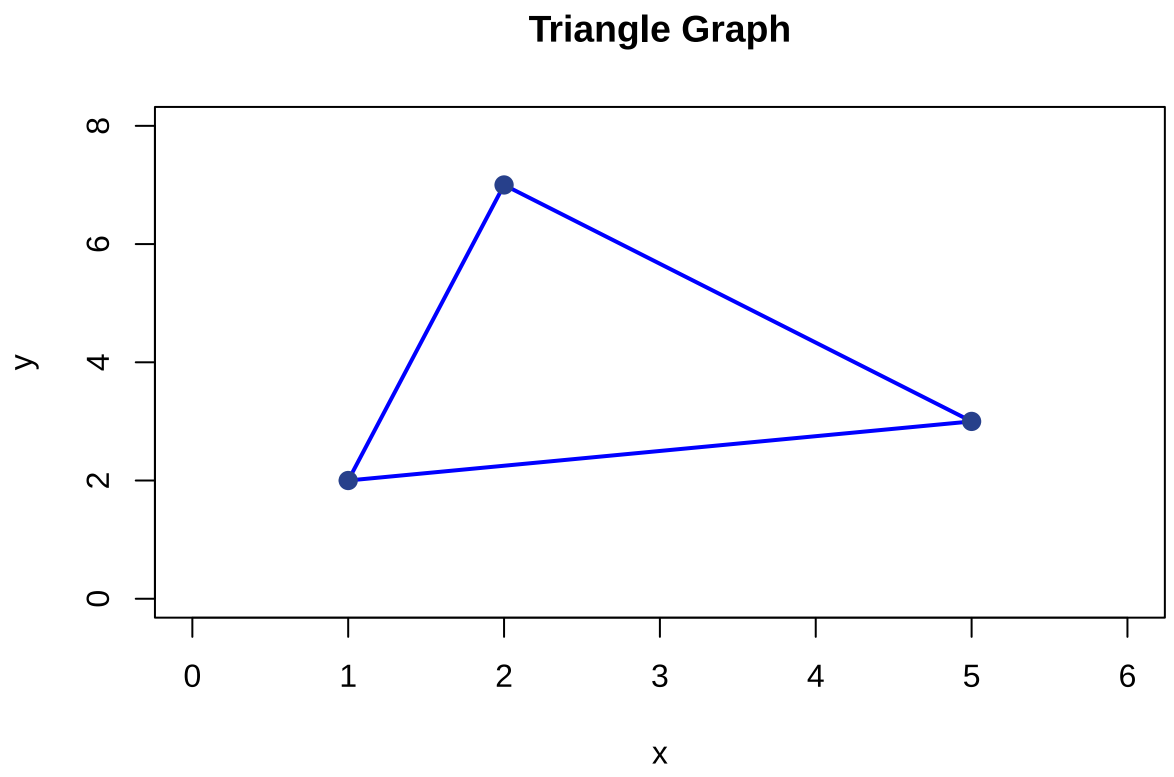


Part (b): Using the `lines()` function

Now we'll draw the same triangle again, with the same points for the vertices, but this time we'll use the `lines()` function. Remember that this function always draws *connected* line segments. Remember to connect the last point C to the first point A! The lines of the triangle should be colored blue with a line width of 2. As in part (a), you should also draw points at the vertices; here you are welcome to copy and paste your code from part (a). Start with the same empty plot with no data that you used in part (a).

Solution

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

plot(
    x = NULL,
    xlim = c(0, 6),
    ylim = c(0, 8),
    main = "Triangle Graph",
    xlab = "x",
    ylab = "y"
)


# Here's where we draw the lines, using the special
# form for the lines() function.

lines(
    x = c(1, 2, 5, 1),
    y = c(2, 7, 3, 2),
    lwd = 2,
    lty = "solid",
    col = "blue"
)


# Now we can draw the points:

points(
    x = 1,
    y = 2,
    pch = 19,
    cex = 1.2,
    col = "royalblue4"
)

points(
    x = 2,
    y = 7,
    pch = 19,
    cex = 1.2,
    col = "royalblue4"
)

points(
    x = 5,
    y = 3,
    pch = 19,
    cex = 1.2,
    col = "royalblue4"
)
```



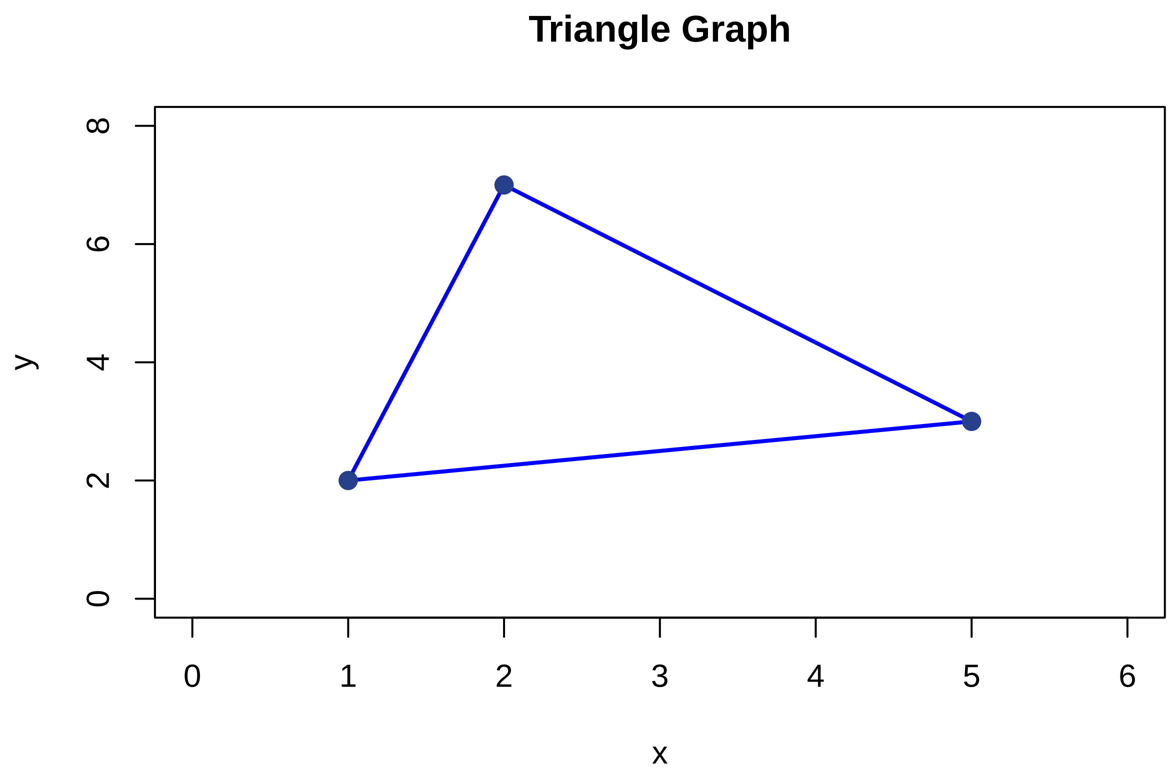
Part (c): Using polygons

Finally, draw the triangle using the `polygon()` function. Remember with this function that it draws connected line segments, and that the last point C is automatically connected to the first point A. The lines of the triangle should be colored blue with a line width of 2. Decorate your triangle with points just like in parts (a) and (b).

Solution

```
# First, we'll create an empty plot with no data:

plot(
    x = NULL,
    xlim = c(0, 6),
    ylim = c(0, 8),
    main = "Triangle Graph",
    xlab = "x",
    ylab = "y"
)


# Now we can use the polygon() function:

polygon(
    x = c(1, 2, 5),
```

```r
    y = c(2, 7, 3),
    lwd = 2,
    lty = "solid",
    border = "blue",
    col = "white"
)


# Now we can draw the points:

points(
    x = 1,
    y = 2,
    pch = 19,
    cex = 1.2,
    col = "royalblue4"
)

points(
    x = 2,
    y = 7,
    pch = 19,
    cex = 1.2,
    col = "royalblue4"
)

points(
    x = 5,
    y = 3,
    pch = 19,
    cex = 1.2,
    col = "royalblue4"
)
```


Triangle Graph
y
x
0
2
4
6
8
0
1
2
3
4
5
6

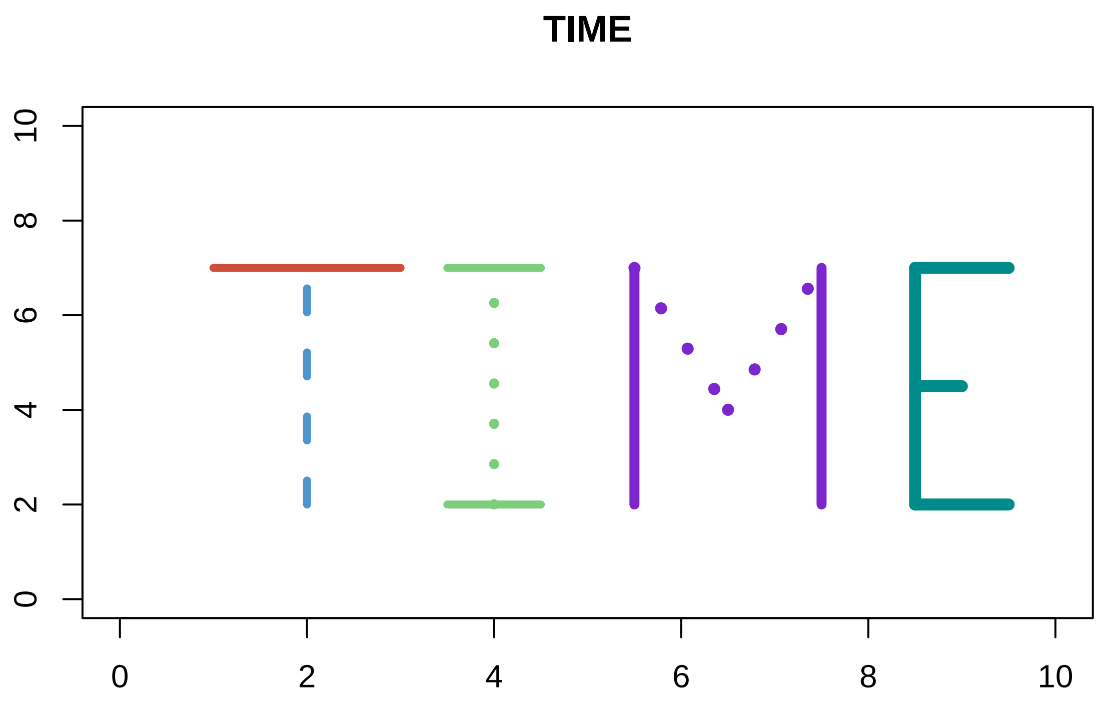
End of Problem 1

Problem 2: TIME

Create an empty plot with no data. You'll have to decide on the ranges of the x- and y-axes. Use "TIME" for the main title, but just use empty strings for the x- and y-axis titles. Then, using only `segments()` commands, draw the letters of the word "TIME" (by my count, you'll need 13 separate 'segments() commands). Try to do something creative with the line type, width, and color. You'll get full credit as long as we can see the letters for the word "TIME", but you'll get more out of the problem if you take an extra 10 minutes to explore this a little bit.

Solution

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




segments(
    x0 = 2,
    y0 = 2,
    x1 = 2,
    y1 =7,
    lty = "dashed",
    lwd = 4,
    col = "steelblue3"
)

segments(
    x0 = 1,
    y0 = 7,
    x1 = 3,
    y1 = 7,
    lty = "solid",
    lwd = 4,
    col = "tomato3"
)

segments(
    x0 = 4,
    y0 = 2,
    x1 = 4,
    y1 = 7,
    lty = "dotted",
    lwd = 5,
    col = "palegreen3"
)

segments(
    x0 = 3.5,
    y0 = 7,
    x1 = 4.5,
    y1 = 7,
    lty = "solid",
    lwd = 4,
    col = "palegreen3"
)

segments(
    x0 = 3.5,
    y0 = 2,
    x1 = 4.5,
    y1 = 2,
    lty = "solid",
    lwd = 4,
    col = "palegreen3"
)

segments(
    x0 = 5.5,
    y0 = 2,
    x1 = 5.5,
    y1 = 7,
    lty = "solid",
    lwd = 5,
    col = "purple3"
)

segments(
    x0 = 5.5,
    y0 = 7,
    x1 = 6.5,
    y1 = 4,
    lty = "dotted",
    lwd = 6,
    col = "purple3"
)

segments(
    x0 = 6.5,
    y0 = 4,
    x1 = 7.5,
    y1 = 7,
    lty = "dotted",
    lwd = 6,
    col = "purple3"
)

segments(
    x0 = 7.5,
    y0 = 2,
    x1 = 7.5,
    y1 = 7,
    lty = "solid",
    lwd = 5,
    col = "purple3"
)


segments(
    x0 = 8.5,
    y0 = 2,
    x1 = 8.5,
    y1 = 7,
    lty = "solid",
    lwd = 6,
    col = "cyan4"
)


segments(
    x0 = 8.5,
    y0 = 7,
    x1 = 9.5,
    y1 = 7,
    lty = "solid",
    lwd = 6,
    col = "cyan4"
)


segments(
    x0 = 8.5,
    y0 = 4.5,
    x1 = 9,
    y1 = 4.5,
    lty = "solid",
    lwd = 6,
    col = "cyan4"
)


segments(
    x0 = 8.5,
    y0 = 2,
    x1 = 9.5,
    y1 = 2,
    lty = "solid",
    lwd = 6,
    col = "cyan4"
)
```


TIME
0
2
4
6
8
10
0
2
4
6
8
10

End of problem 2

Problem 3: Sales Chart

A grocery store has data on the sales of three breakfast cereals: Sugar Bomz (SBZ), Krispee Yummm (KYM), and Healthy Kale and Tofu (HKT):

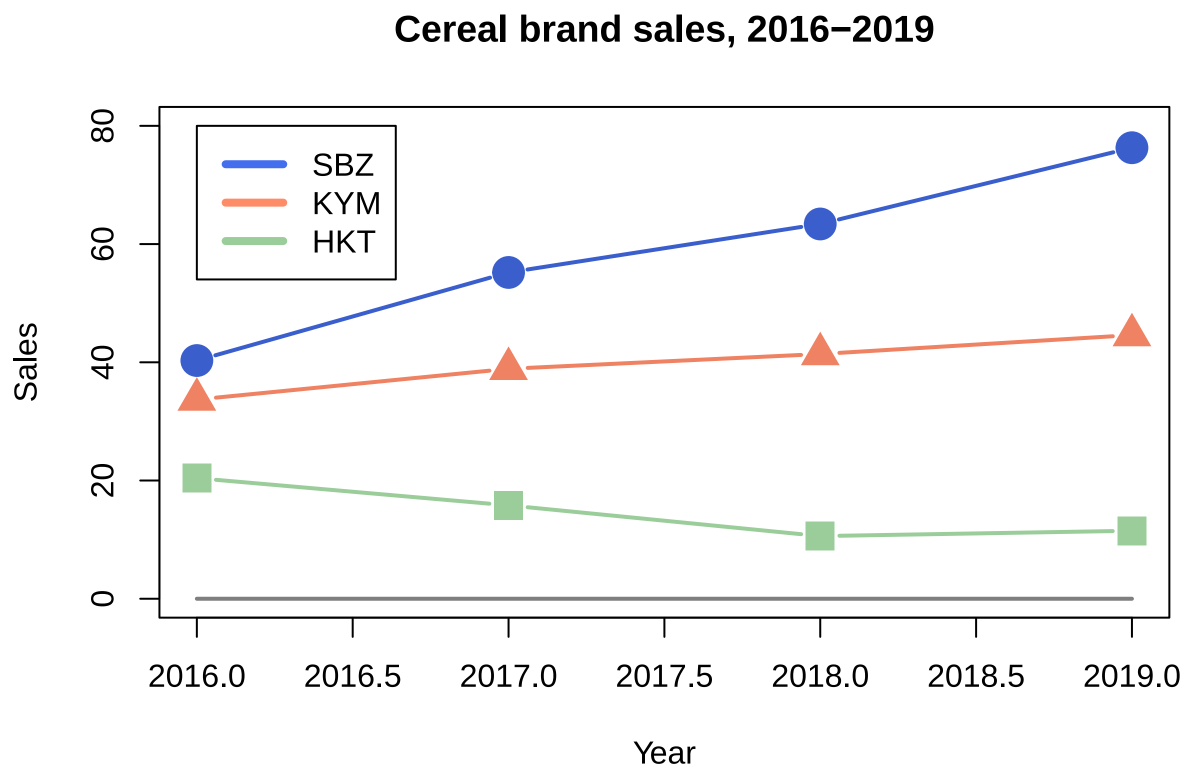
Year	SBZ	KYM	HKT
2016	40.3	33.7	20.4
2017	55.2	38.9	15.8
2018	63.4	41.4	10.6
2019	76.3	44.6	11.5

Create a sales chart for this data. The x-axis should indicate the year, and the y-axis should indicate the sales volume. Draw the sales for each brand as a connected line, and use a different point shape and color for each brand. Remember to include a main title, axis titles, and a legend.

Solution

```
plot(
    x = NULL,
    xlim = c(2016, 2019),
    ylim = c(0, 80),
    main = "Cereal brand sales, 2016-2019",
    xlab = "Year",
    ylab = "Sales"
)

# Draw a horizontal reference line:

segments(
    x0 = 2016,
    y0 = 0,
    x1 = 2019,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Now draw the line graphs:

lines(
    x = c(2016, 2017, 2018, 2019),
    y = c(40.3, 55.2, 63.4, 76.3),
    type = "b",
    lty = "solid",
    lwd = 2,
    col = "royalblue3",
    pch = 19,
    cex = 2
)
```

```
lines(
    x = c(2016, 2017, 2018, 2019),
    y = c(33.7, 38.9, 41.4, 44.6),
    type = "b",
    lty = "solid",
    lwd = 2,
    col = "salmon2",
    pch = 17,
    cex = 2
)


lines(
    x = c(2016, 2017, 2018, 2019),
    y = c(20.4, 15.8, 10.6, 11.5),
    type = "b",
    lty = "solid",
    lwd = 2,
    col = "darkseagreen3",
    pch = 15,
    cex = 2
)

legend(
    x = 2016,
    y = 80,
    legend = c( "SBZ", "KYM", "HKT"),
    lty = "solid",
    lwd = 4,
    col =
        c( "royalblue2", "salmon1", "darkseagreen3")
)
```


Cereal brand sales, 2016−2019
SBZ
KYM
HKT
Sales
0
20
40
60
80
2016.0
2016.5
2017.0
2017.5
2018.0
2018.5
2019.0
Year

End of problem 3

Problem 4: Graphing Polynomials

In this problem, we will gain experience graphing polynomial functions.

Part (a): Quadratic Functions

First, let's graph this function:

$$y \;=\; 2x^2 - 2x - 4$$

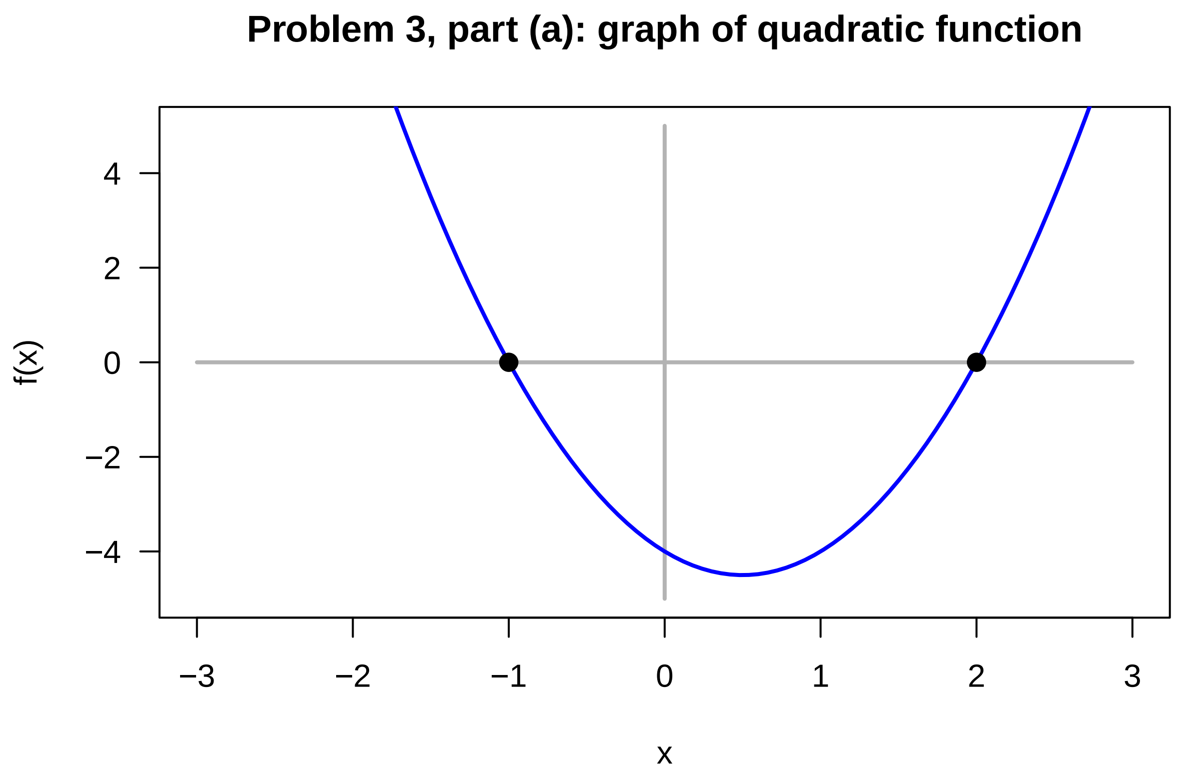
Make your graph using these steps:

- First, create an empty plot with the x-axis ranging from -3 to +3 and the y-axis ranging from -5 to +5. Label the x axis with "x" and the y axis with "f(x)".
- Draw in a horizontal reference line $y = 0$ from $x = -3$ to $x = +3$.
- Draw in a vertical reference line $x = 0$ from $y = -5$ to $y = +5$.
- Draw the graph of the function using the `curve()` command. (Hint: don't forget to use the `add = TRUE` option!) The function curve should be a solid line of width 2, and should be colored blue.
- Draw points at the locations $(-1, 0)$ and $(2, 0)$, which represent the roots of the quadratic function.

Solution

```
# First, create an empty plot

plot(
    x = NULL,
    xlim = c(-3, 3),
    ylim = c(-5, 5),
    main = "Problem 3, part (a): graph of quadratic function",
    xlab = "x",
    ylab = "f(x)",
    las = 1
)


# Draw the horizontal and vertical axes

segments(
    x0 = -3,
    y0 = 0,
    x1 = 3,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray70"
)


segments(
    x0 = 0,
    y0 = -5,
```

```r
    x1 = 0,
    y1 = 5,
    lty = "solid",
    lwd = 2,
    col = "gray70"
)


# Now we can draw the graph of the curve. The problem statement required
# the line to be solid, so we use the lty = 1 option. The width of the line
# should be 2, so we use the lwd = 2 option. The color of the line should be
# blue, so we use the col = "blue" option. Finally, we have to use the
# add = TRUE option so that the curve is drawn on the pre-existing plot,
# not a new plot.

curve(
  expr = 2 * x^2 - 2 * x - 4,
  lty = 1,
  lwd = 2,
  col = "blue",
  add = TRUE
)


# Add the points

points(
  x = -1,
  y = 0,
  pch = 19,
  cex = 1.2
)

points(
  x = 2,
  y = 0,
  pch = 19,
  cex = 1.2
)
```



Part (b): Cubic functions

Now let's draw the graph of a cubic equation:

$$y \;=\; 2x^3 - 2x$$

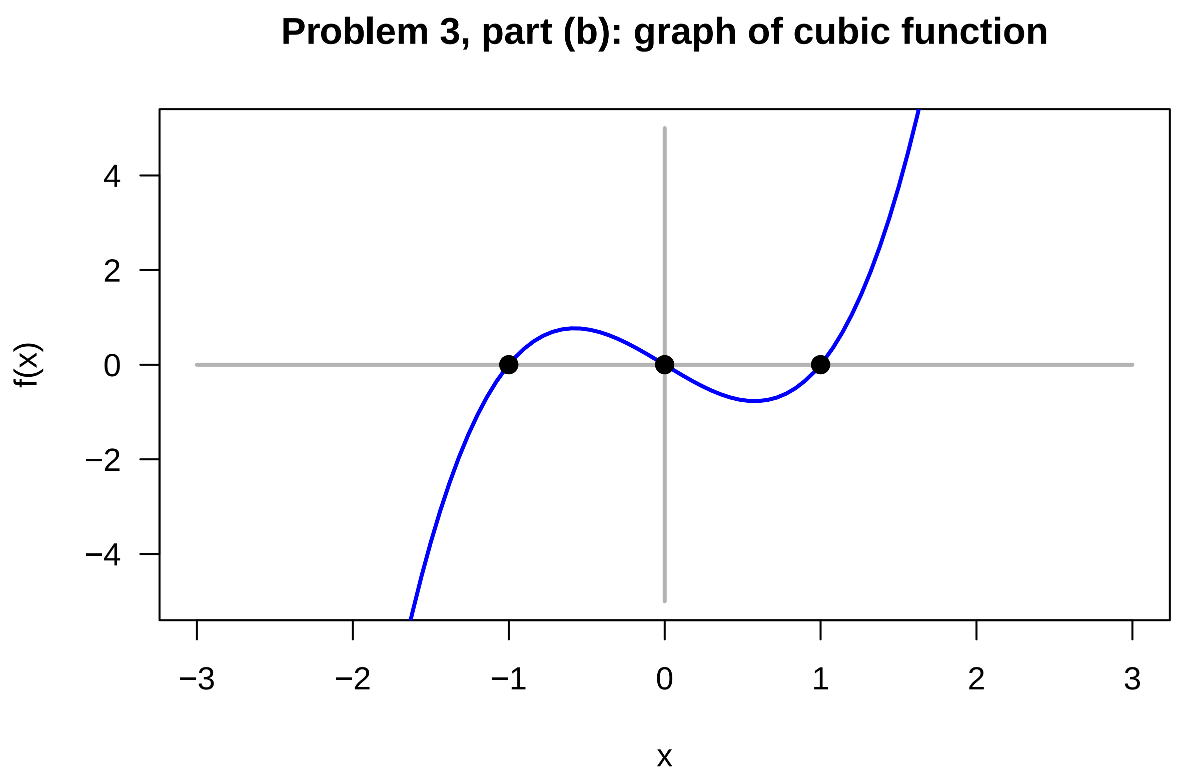
Make your graph using these steps:

- First, create an empty plot with x ranging from -3 to +3 and y ranging from -5 to +5. Label the x axis with "x" and the y axis with "f(x)".
- Draw in horizontal and vertical reference lines.
- Using a solid blue line of width 3, draw the graph of the function using the `curve()` command.
- Draw points at the locations (-1, 0), (0, 0), and (1, 0), which represent the roots of the cubic function.

Solution

```
# Create an empty plot with no data

plot(
    x = NULL,
    xlim = c(-3, 3),
    ylim = c(-5, 5),
    main = "Problem 3, part (b): graph of cubic function",
    xlab = "x",
    ylab = "f(x)",
    las = 1
)


# Draw the horizontal and vertical reference lines:

segments(
    x0 = -3,
    y0 = 0,
    x1 = 3,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray70"
)

segments(
    x0 = 0,
    y0 = -5,
    x1 = 0,
    y1 = 5,
    lty = "solid",
    lwd = 2,
    col = "gray70"
)


# Now we can draw the graph of the curve. The problem statement required
# the line to be solid, so we use the lty = 1 option. The width of the line
# should be 2, so we use the lwd = 2 option. The color of the line should be
# blue, so we use the col = "blue" option. Finally, we have to use the
# add = TRUE option so that the curve is drawn on the pre-existing plot,
# not a new plot.

curve(
    expr = 2 * x^3 -  2 * x,
    lty = "solid",
    lwd = 2,
    col = "blue",
    add = TRUE
)


# Finally, we can draw in the three points:

points(
    x = -1,
    y = 0,
    pch = 19,
    cex = 1.2,
    col = "black"
```

```
)

points(
    x = 0,
    y = 0,
    pch = 19,
    cex = 1.2,
    col = "black"
)

points(
    x = 1,
    y = 0,
    pch = 19,
    cex = 1.2,
    col = "black"
)
```



End of Problem 4

Problem 5: Exponential Functions

Create an empty plotting region with no data, where the x-axis ranges from 0 to 4 and the y-axis ranges from 0 to 2. Include horizontal and vertical reference lines. Also, draw in the dashed horizontal line $y = 1$. Then plot the curves for these functions:

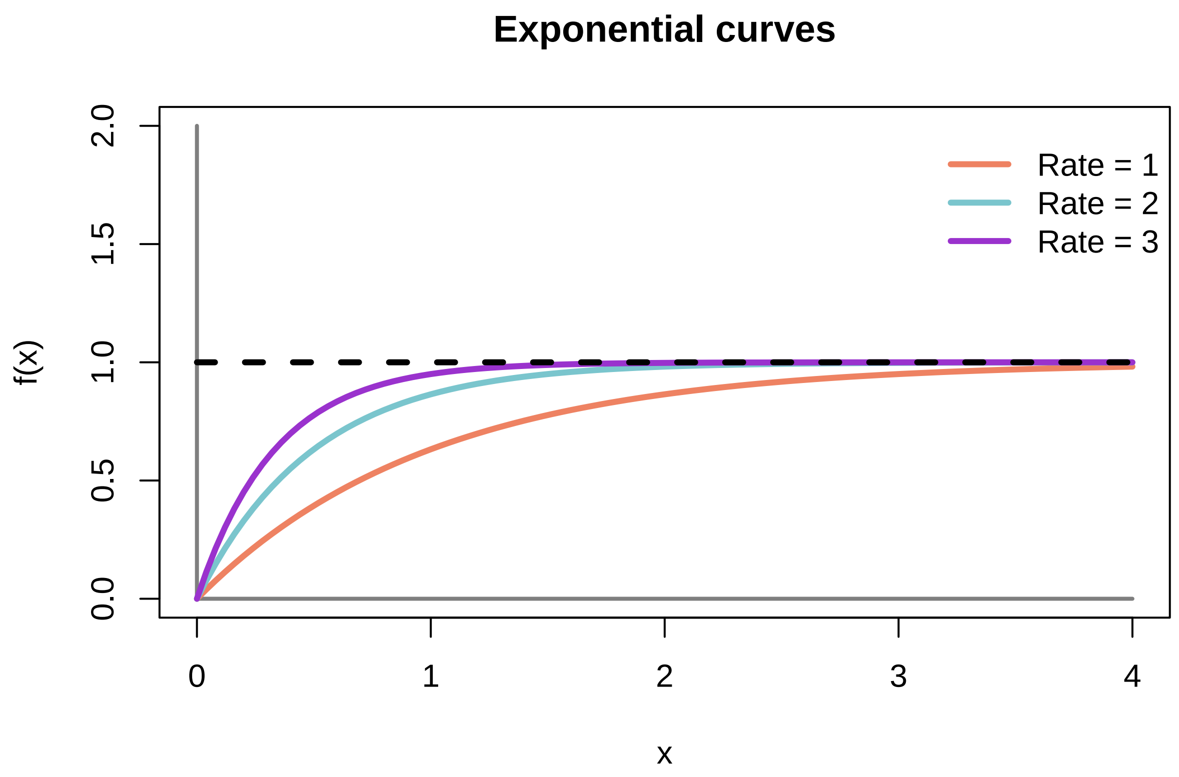
- The function $f(x) = 1 - e^{-x}$.
- The function $g(x) = 1 - e^{-2x}$.
- The function $h(x) = 1 - e^{-3x}$.

Finally, include a main title, axis titles, and a legend.

Solution

```
plot(
    x = NULL,
    xlim = c(0, 4),
    ylim = c(0, 2),
    main = "Exponential curves",
    xlab = "x",
    ylab = "f(x)"
)

segments(
    x0 = 0,
    y0 = 0,
    x1 = 4,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


segments(
    x0 = 0,
    y0 = 0,
    x1 = 0,
    y1 = 2,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)




curve(
    expr = 1 - exp(-x),
    lty = "solid",
    lwd = 3,
    col = "salmon2",
```

```r
    add = TRUE
)


curve(
    expr = 1 - exp(-2*x),
    lty = "solid",
    lwd = 3,
    col = "cadetblue3",
    add = TRUE
)


curve(
    expr = 1 - exp(-3*x),
    lty = "solid",
    lwd = 3,
    col = "darkorchid3",
    add = TRUE
)


segments(
    x0 = 0,
    y0 = 1,
    x1 = 4,
    y1 = 1,
    lty = "dashed",
    lwd = 3,
    col = "black"
)



legend(
    x = 3.1,
    y = 2,
    legend =
        c( "Rate = 1", "Rate = 2", "Rate = 3"),
    lty = "solid",
    lwd = 3,
    col = c( "salmon2", "cadetblue3", "darkorchid3"),
    bty = "n"
)
```


Exponential curves
f(x)
x
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
Rate = 1
Rate = 2
Rate = 3

End of problem 5

Problem 6: Mondrian

Time to express our inner souls with some more modern art!

Go to the Wikipedia article on Piet Mondrian, the great Dutch abstract artist, and scroll down to find the image of *Composition II in Red, Blue, and Yellow*, painted in 1930. Your goal is to recreate this painting using the tools that we developed in lecture.

- First, create a completely blank square plot with no axes, with x ranging from 0 to 10 and y ranging from 0 to 10.
- Draw a frame around the plot area by drawing a polygon with vertices at (0, 0), (0, 10), (10, 10), and (10, 0).
- Draw the red, blue, and yellow rectangles using the `polygon()` function. You'll have to figure out the appropriate coordinates!
- Finally, draw in the black lines, using the `lwd =` option to control the width of the lines.

You'll find it useful when drawing thick lines to include this line code at the beginning of the code chunk:

```
par( lend = 1)
```

Your picture doesn't have to be perfect, but it should be a reasonable approximation to the original.

Solution

```
# This command makes a square plot

par( pty = "s" )

# This command removes the borders

par( bty = "n" )

# This command controls how the line segments end

par( lend = 1)


# Create an empty plot:

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


# Put a border around the plot

polygon(
    x = c(0, 0, 10, 10),
    y = c(0, 10, 10, 0),
    lty = "solid",
    lwd = 1,
    border = "black"
)


# Draw the red, yellow, and blue shapes

polygon(
    x = c(3, 3, 10, 10 ),
    y = c( 3, 10, 10, 3 ),
    col = "red"
)

polygon(
    x = c(9, 9, 10, 10 ),
    y = c( 0, 1.5, 1.5, 0 ),
    col = "yellow"
)

polygon(
    x = c(0, 0, 3, 3 ),
    y = c( 0, 3, 3, 0 ),
    col = "blue"
)


# Draw the line segments

segments(
    x0 = 3,
    y0 = 0,
    x1 = 3,
    y1 = 10,
    lty = "solid",
    lwd = 8,
    col = "black"
)

segments(
    x0 = 0,
    y0 = 3,
    x1 = 10,
    y1 = 3,
    lty = "solid",
    lwd = 8,
    col = "black"
)

segments(
    x0 = 0,
```

```r
    y0 = 7,
    x1 = 3,
    y1 = 7,
    lty = "solid",
    lwd = 15,
    col = "black"
)

segments(
    x0 = 9,
    y0 = 0,
    x1 = 9,
    y1 = 3,
    lty = "solid",
    lwd = 8,
    col = "black")

segments(
    x0 = 9,
    y0 = 1.5,
    x1 = 10,
    y1 = 1.5,
    lty = "solid",
    lwd = 8,
    col = "black"
)
```