Problem Set 2 Solutions

CSCI E-5a: Programming in R

Let's clear the global computing environment:

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

Problem 1: Baseball Statistics

The great player Willie Mays had these career statistics:

Statistics	Value
Plate appearances	12,497
At-bats	10,881
Hits	3,283
Doubles	523
Triples	140
Home Runs	660
Bases on balls	1,464
Hit by a pitch	44
Sacrifice flies	91

Part (a)

Calculate Willie Mays' batting average. Report your result using a `cat()` statement, displaying this value with 3 decimal places.

Solution

```
hits <- 3283

at.bats <- 10881

batting.average <-
    hits / at.bats

cat(
    "Willie Mays batting average:",
    formatC(
        batting.average,
        format = "f",
        digits = 3
    )
)
```

```
## Willie Mays batting average: 0.302
```

Part (b)

Calculate the number of singles that Mays hit. Report your result using a `cat()` statement.

Solution

```
doubles <- 523

triples <- 140

home.runs <- 660

singles <-
    hits -
    (doubles + triples + home.runs)

cat(
    "Willie Mays total singles:",
    formatC(
        singles,
        format = "f",
        big.mark = ",",
        digits = 0
    )
)
```

```
## Willie Mays total singles: 1,960
```

Part (c)

Calculate the total number of outs for Mays. Report your result using a `cat()` statement.

Solution

```
outs <- at.bats - hits

cat(
    "Willie Mays total outs:",
    formatC(
        outs,
        format = "f",
        big.mark = ",",
        digits = 0
    )
)
```

```
## Willie Mays total outs: 7,598
```

Part (d)

Calculate the career on-base percentage (OBP) for Mays. Report your result using a `cat()` statement, formatting this value with 3 decimal places.

Solution

```r
bases.on.balls <- 1464

hit.by.pitch <- 44

sacrifice.flies <- 91

on.base.percentage <-
  (hits + bases.on.balls + hit.by.pitch) /
  (at.bats + bases.on.balls + hit.by.pitch + sacrifice.flies)

cat(
  "On-base percentage:",
  formatC(
    x = on.base.percentage,
    format = "f",
    digits = 3
  )
)
```

```
## On-base percentage: 0.384
```

Part (e)

Calculate the number of total bases for Mays. Report your result using a `cat()` statement.

Solution

```r
total.bases <-
  singles +
  (2 * doubles) +
  (3 * triples) +
  (4 * home.runs)

cat(
  "Total bases:",
  formatC(
    x = total.bases,
    format = "f",
    digits = 0
  )
)
```

```
## Total bases: 6066
```

Part (f)

Calculate the career slugging percentage (SLG) for Mays. Report your result using a `cat()` statement, formatting this value with 3 decimal places.

Solution

```
slugging.percentage <-
  total.bases / at.bats

cat(
  "Slugging percentage:",
  formatC(
    x = slugging.percentage,
    format = "f",
    digits = 3
  )
)
```

```
## Slugging percentage: 0.557
```

Part (g)

Construct a pie chart showing the relative proportions of these values for Willie Mays:

- Singles (1B)
- Doubles (2B)
- Triples (3B)
- Home Runs (HR)
- Bases on Balls (BB)
- Hit by a pitch (HBP)
- Outs
- Everything else

Use clockwise rotation for the pie chart, and be sure to provide a main title.

Solution

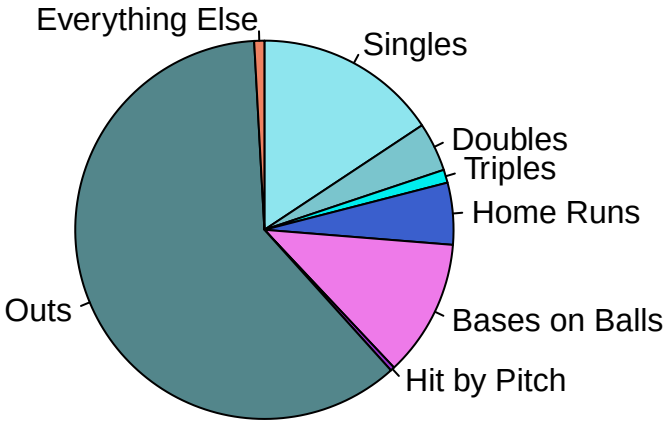
```
cat(
  4,
  ", ",
  5,
  sep = ""
)
```

```
## 4, 5
```

```
plate.appearances <- 12497

everything.else <-
  plate.appearances -
  (singles + doubles + triples + home.runs +
     bases.on.balls + hit.by.pitch + outs)
```

```r
data.vector <-
  c(
    singles,
    doubles,
    triples,
    home.runs,
    bases.on.balls,
    hit.by.pitch,
    outs,
    everything.else
  )

labels.vector <-
  c(
    "Singles",
    "Doubles",
    "Triples",
    "Home Runs",
    "Bases on Balls",
    "Hit by Pitch",
    "Outs",
    "Everything Else"
  )

colors.vector <-
  c(
    "cadetblue2",
    "cadetblue3",
    "cyan2",
    "royalblue3",
    "orchid2",
    "darkorchid3",
    "cadetblue4",
    "salmon2"
  )

pie(
  x = data.vector,
  labels = labels.vector,
  main = "Pie chart of Willie Mays Plate Appearances",
  clockwise = TRUE,
  col = colors.vector
)
```

Pie chart of Willie Mays Plate Appearances



End of problem 1

Problem 2: Calculating Final Grades

Ashley is registered for graduate credit, and at the end of term she has these results:

- Total problem set points raw score: 58
- Midterm raw score: 69
- Final comprehensive assessment raw score: 76

Part (a)

Calculate the problem set standardized score for Ashley. Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Solution

Let's start by defining some variables:

```
total.problem.set.points.raw.score <- 58

midterm.exam.raw.score <- 69

comprehensive.course.assessment.raw.score <- 76
```

First we calculate the problem set standardized score:

```
problem.set.standardized.score <-
    total.problem.set.points.raw.score / 68 * 100

cat( "Problem set standardized score:",
     formatC(
         problem.set.standardized.score,
         format = "f",
         digits = 2
     )
)
```

```
## Problem set standardized score: 85.29
```

Part (b)

Calculate the midterm exam standardized score for Ashley. Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Solution

Now we can standardize the midterm exam score:

```
midterm.exam.standardized.score <-
    midterm.exam.raw.score / 80 * 100

cat( "Midterm exam standardized score:",
```

```
    formatC(
        midterm.exam.standardized.score,
        format = "f",
        digits = 2
    )
)
```

```
## Midterm exam standardized score: 86.25
```

Part (c)

Calculate the comprehensive course assessment standardized score for Ashley. Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Solution

Now we can standardize the comprehensive course assessment score:

```
comprehensive.course.assessment.standardized.score <-
    comprehensive.course.assessment.raw.score / 80 * 100

cat( "Comprehensive course assessment standardized score:",
     formatC(
         comprehensive.course.assessment.standardized.score,
         format = "f",
         digits = 2
     )
)
```

```
## Comprehensive course assessment standardized score: 95.00
```

Part (d)

Calculate the preliminary score 1 for Ashley, using a weighted average of the problem set, midterm, and comprehensive course assessment standardized scores. Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Solution

```
preliminary.score.1 <-
    (0.2 * problem.set.standardized.score) +
    (0.3 * midterm.exam.standardized.score) +
    (0.5 * comprehensive.course.assessment.standardized.score)

cat( "Preliminary score 1:",
     formatC(
         preliminary.score.1,
         format = "f",
         digits = 2
     )
)
```

```
## Preliminary score 1: 90.43
```

Part (e)

Calculate the preliminary score 2 for Ashley, using a weighted average of the midterm and comprehensive course assessment standardized scores. Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Solution

```
preliminary.score.2 <-
    (0.4 * midterm.exam.standardized.score) +
    (0.6 * comprehensive.course.assessment.standardized.score)

cat( "Preliminary score 2:",
     formatC(
         preliminary.score.2,
         format = "f",
         digits = 2
     )
)
```

```
## Preliminary score 2: 91.50
```

Part (f)

Based on your results from parts (d) and (e), what is Ashley's final graduate course score? Explain your answer using a few sentences.

Solution

The final score is the maximum of the two preliminary scores. Since the preliminary score 1 was 90.43 and the preliminary score 2 was 91.50, the maximum of the two scores is 91.50, and this is the final graduate course score.

```
graduate.final.course.score <-
    max( preliminary.score.1, preliminary.score.2 )

cat(
    "Graduate final course score:",
    formatC(
        graduate.final.course.score,
        format = "f",
        digits = 2
    )
)
```

```
## Graduate final course score: 91.50
```

Part (g)

Determine the final course letter grade for Ashley. Explain your answer with a few sentences.

Solution

Ashley's final graduate course score is 91.50, and this is contained in the range [90, 93), which is the range for an "A-". Therefore Ashley receives an "A-" for the final course letter grade.

Part (h)

Miles has the same raw scores as Ashley does, but is registered for undergraduate credit. Report the undergraduate final course score for Miles using a `cat()` statement, displaying this value to 2 decimal places.

Solution

```
undergraduate.final.course.score <-
    4/3 * graduate.final.course.score

cat(
    "Miles' final course score:",
    formatC(
        undergraduate.final.course.score,
        format = "f",
        digits = 2
    )
)
```

```
## Miles' final course score: 122.00
```

Part (i)

What is Miles' final course letter grade? Explain your answer with a complete sentence.

Solution

Miles' undergraduate final course score is 122.00, and this is well above the threshold value of 93 for an "A". So Miles' final course letter grade is an "A".

End of problem 2

Problem 3: Quadratic Functions

Part (a)

Consider the quadratic function $f(x) = x^2 - 6x + 9$. What is the value of the discriminant for this function? Report your answer using a `cat()` statement, formatting the value to 2 decimal places.

Solution

Let's first define some variables:

```
a <- 1

b <- -6

c <- 9
```

Now we can calculate the discriminant:

```
discriminant <-
    b^2 - 4 * a * c

cat(
    "Discriminant:",
    formatC(
        discriminant,
        format = "f",
        digits = 2
    )
)
```

```
## Discriminant: 0.00
```

Part (b)

Based on your answer to part (a), how many solutions exist for the equation $f(x) = 0$? Report your answer with one or two sentences.

Solution*

The value of the discriminant is 0, so that means that there is exactly 1 root of this quadratic polynomial.

Part (c)

Use the quadratic formula, determine all the roots of the equation $f(x) = 0$. Report each value using a `cat()` statment, formatting the value to 2 decimal places.

Solution

Here we use the standard formula:

```
only.root <-
    - b / (2 * a)
```

Now let's report this value:

```
cat(
    "Only root:",
    formatC(
        only.root,
        format = "f",
        digits = 2
    )
)
```

```
## Only root: 3.00
```

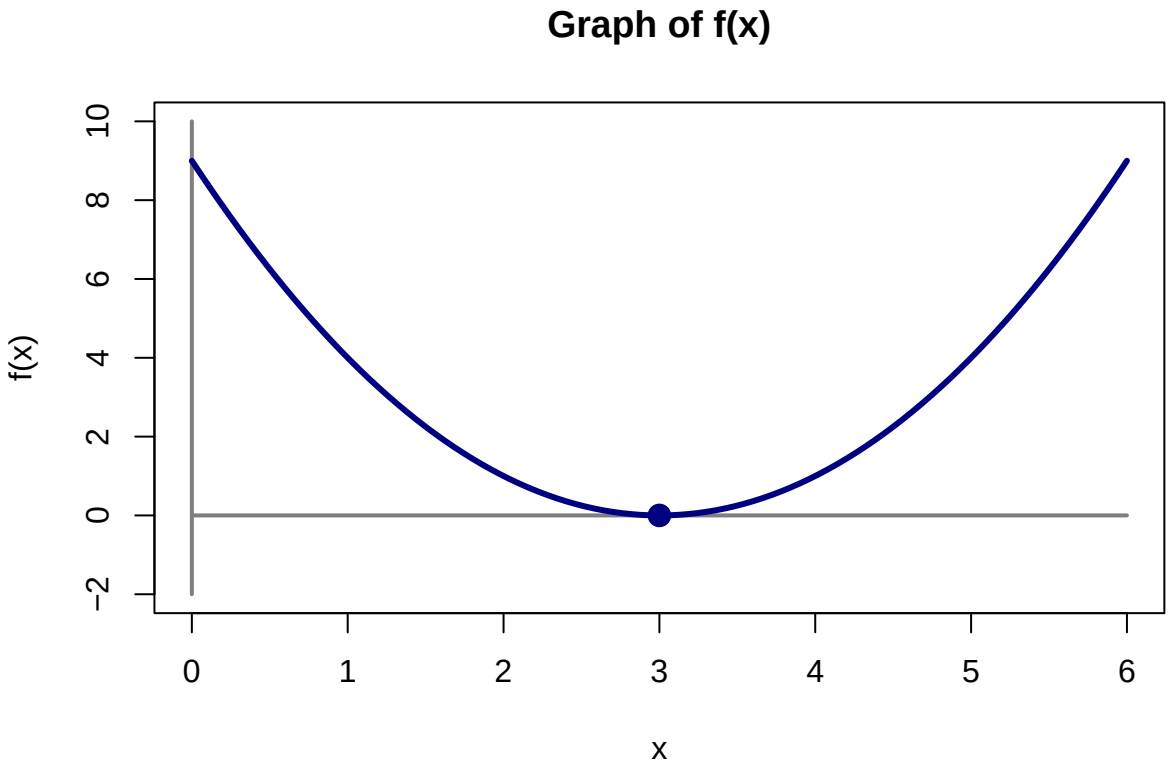
Part (d)

Draw a graph of $f(x)$. The x-values for the plotting region should range from 0 to 6, and the y-values should range from -2 to +8. Include horizontal and vertical reference lines. Draw the curve of $f(x)$, explicitly selecting the line type, width, and color, and then draw in points for any roots of the equation, again explicitly selecting the type, size, and color. Be sure to provide a main title and titles for the x-axis and the y-axis.

Solution

```
plot(
    x = NULL,
    xlim = c(0, 6),
    ylim = c(-2, 10),
    main = "Graph of f(x)",
    xlab = "x",
    ylab = "f(x)"
)


# Draw the horizontal reference line

segments(
    x0 = 0,
    y0 = 0,
    x1 = 6,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Draw the vertical reference line

segments(
    x0 = 0,
    y0 = -2,
    x1 = 0,
    y1 = 10,
```

```
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Draw the curve of f(x)

curve(
    x^2 - 6 * x + 9,
    lty = "solid",
    lwd = 3,
    col = "navy",
    add = TRUE
)


# Draw the only root of the equation

points(
    x = 3,
    y = 0,
    pch = 19,
    cex = 1.5,
    col = "navy"
)
```



Part (e)

Consider the quadratic function $g(x) = x^2 - 10x + 21$. What is the value of the discriminant for this function? Report your answer using a `cat()` statement, formatting the value to 2 decimal places.

Solution

Let's first define some variables:

```
a <- 1

b <- -10

c <- 21
```

Now we can calculate the discriminant:

```
discriminant <-
    b^2 - 4 * a * c

cat(
    "Discriminant:",
    formatC(
        discriminant,
        format = "f",
        digits = 2
```

```
    )
)
```

```
## Discriminant: 16.00
```

Part (f)

Based on your answer to part (e), how many solutions exist for the equation $g(x) = 0$? Report your answer with one or two sentences.

Solution

The discriminant has a value that is strictly greater than 0, thus the quadratic polynomial has 2 roots.

Part (g)

Use the quadratic formula, determine all the roots of the equation $g(x) = 0$. Report each value using a `cat()` statment, formatting the value to 2 decimal places. If there are no solutions, write a short stating that.

Solution

Let's calculate the first root:

```
first.root <-
    (-b - sqrt( b^2 - 4 * a * c ) ) / (2 * a)

cat(
    "The first root of the function is:",
    formatC(
        first.root,
        format = "f",
        digits = 2
    )
)
```

```
## The first root of the function is: 3.00
```

Now let's calculate the second root:

```
second.root <-
    (-b + sqrt( b^2 - 4 * a * c ) ) / (2 * a)

cat(
    "The second root of the equation is:",
    formatC(
        second.root,
        format = "f",
        digits = 2
    )
)
```

```
## The second root of the equation is: 7.00
```

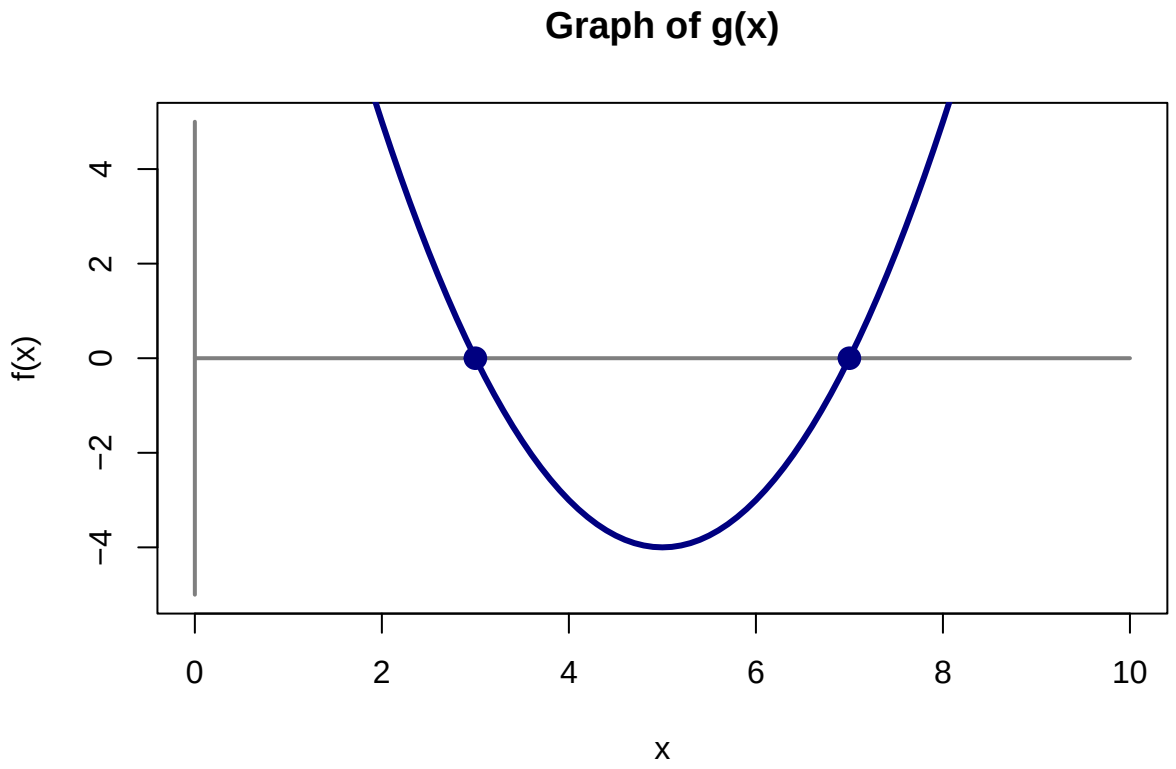
Part (h)

Draw a graph of $g(x)$. The x-values for the plotting region should range from 0 to 10, and the y-values should range from -5 to +5. Include horizontal and vertical reference lines. Draw the curve of $g(x)$, explicitly selecting the line type, width, and color, and then draw in points for any roots of the equation, again explicitly selecting the type, size, and color. Be sure to provide a main title and titles for the x-axis and the y-axis.

Solution

```
plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(-5, 5),
    main = "Graph of g(x)",
    xlab = "x",
    ylab = "f(x)"
)


# Draw the horizontal reference line

segments(
    x0 = 0,
    y0 = 0,
    x1 = 10,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Draw the vertical reference line

segments(
    x0 = 0,
    y0 = -5,
    x1 = 0,
    y1 = 5,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Draw the curve of f(x)

curve(
    x^2 - 10 * x + 21,
    lty = "solid",
    lwd = 3,
    col = "navy",
    add = TRUE
)
```

```
# Draw the roots of the equation

points(
    x = c(3, 7),
    y = c(0, 0),
    pch = 19,
    cex = 1.5,
    col = "navy"
)
```



Part (i)

Consider the quadratic function $h(x) = x^2 - 12x + 40$. What is the value of the discriminant for this function? Report your answer using a `cat()` statement, formatting the value to 2 decimal places.

Solution

Let's first define some variables:

```
a <- 1

b <- -12

c <- 40
```

Now we can calculate the discriminant:

```
discriminant <-
    b^2 - 4 * a * c

cat(
    "Discriminant:",
    formatC(
        discriminant,
        format = "f",
        digits = 2
    )
)
```

```
## Discriminant: -16.00
```

Part (j)

Based on your answer to part (a), how many solutions exist for the equation $h(x) = 0$? Report your answer with one or two sentences.

Solution*

Since the value of the discriminant is strictly less than 0, there are no roots of this quadratic polynomial.

Part (k)

Use the quadratic formula, determine all the roots of the equation $h(x) = 0$. Report each value using a `cat()` statment, formatting the value to 2 decimal places.

Solution

There are no roots of this polynomial.

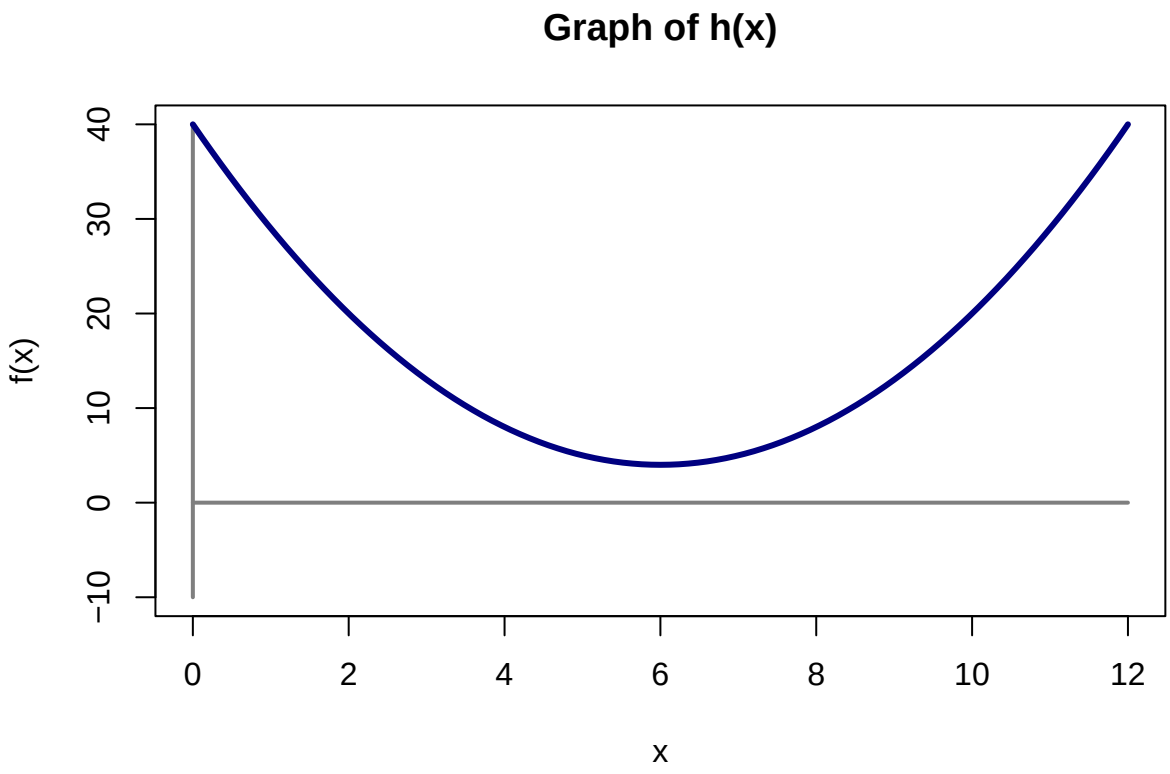
Part (l)

Draw a graph of $h(x)$. The x-values for the plotting region should range from 0 to 12, and the y-values should range from -2 to +20. Include horizontal and vertical reference lines. Draw the curve of $h(x)$, explicitly selecting the line type, width, and color, and then draw in points for any roots of the equation, again explicitly selecting the type, size, and color. Be sure to provide a main title and titles for the x-axis and the y-axis.

Solution

```
plot(
    x = NULL,
    xlim = c(0, 12),
    ylim = c(-10, 40),
    main = "Graph of h(x)",
    xlab = "x",
    ylab = "f(x)"
)
```

```r
# Draw the horizontal reference line

segments(
    x0 = 0,
    y0 = 0,
    x1 = 12,
    y1 = 0,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Draw the vertical reference line

segments(
    x0 = 0,
    y0 = -10,
    x1 = 0,
    y1 = 40,
    lty = "solid",
    lwd = 2,
    col = "gray50"
)


# Draw the curve of f(x)

curve(
    x^2 - 12 * x + 40,
    lty = "solid",
    lwd = 3,
    col = "navy",
    add = TRUE
)
```




```
cat(
  formatC(
    0.300,
    format = "f",
    digits = 3
  )
)
```

```
## 0.300
```

End of problem 3

Problem 4: Grocery Store

A grocery store sells three brands of breakfast cereal:

- Sugar Bomz (SBZ), which costs $2.49 per box.
- Krispee Yummm!! (KYM), which costs $3.99 per box.
- Healthy Kale and Tofu (HKT), which costs $8.99 per box.

The store is currently running a promotional campaign on breakfast cereal by offering a $1 off coupon to customers.

Note that the coupon is applied to the entire order, not to each box of cereal.

On Saturday, the grocery store had six sales:

Transaction ID	Brand	Number of Boxes	Coupon
1	KYM	2	No
2	SBZ	4	Yes
3	SBZ	6	Yes
4	KYM	1	No
5	HKT	2	Yes
6	SBZ	5	Yes

To calculate the net revenue for each transaction, multiply the price per box by the number of boxes and subtract the coupon amount (if any).

To calculate the total net revenue, add up the net revenues for all transactions.

Calculate the total net revenue for the cereal sales for Saturday.

Report this total net revenue using a `cat()` statement, formatting this value with 2 decimal places.

By the way, we're doing this problem to experience how tedious the calculation is.

It's annoying with just 6 transactions, but think how bad it would be if we had 10,000 transactions!

One of the goals of CSCI S-5a is develop techniques that will allow us to automate this sort of calculation.

Solution

There are many different ways to perform this calculation. For these solutions, I'm going to show you an extreme approach to defining variables for every step of the computation. You don't have to do this, and in the end we just want you to tell us what the total profit is.

First, let's create variables for the price per box for each brand of cereal:

```
sbz.price.per.box <- 2.49
kym.price.per.box <- 3.99
hkt.price.per.box <- 8.99

coupon.value <- 1
```

Here are the variables for the price per box of cereal sold in each transaction:

```
transaction.1.price.per.box <- kym.price.per.box

transaction.2.price.per.box <- sbz.price.per.box

transaction.3.price.per.box <- sbz.price.per.box

transaction.4.price.per.box <- kym.price.per.box

transaction.5.price.per.box <- hkt.price.per.box

transaction.6.price.per.box <- sbz.price.per.box
```

Here are the variables for the number of boxes of cereal sold in each transaction:

```
transaction.1.number.of.boxes <- 2

transaction.2.number.of.boxes <- 4

transaction.3.number.of.boxes <- 6

transaction.4.number.of.boxes <- 1

transaction.5.number.of.boxes <- 2

transaction.6.number.of.boxes <- 5
```

Here's the profit for the first transaction:

```
transaction.1.profit <-
  (transaction.1.price.per.box * transaction.1.number.of.boxes)

cat(
  "Transaction 1 profit:",
  formatC(
    transaction.1.profit,
    format = "f",
    digits = 2
  )
)
```

```
## Transaction 1 profit: 7.98
```

Here's the profit for the second transaction:

```
transaction.2.profit <-
  (transaction.2.price.per.box * transaction.2.number.of.boxes) - coupon.value

cat(
  "Transaction 2 profit:",
  formatC(
    transaction.2.profit,
    format = "f",
```

```
    digits = 2
  )
)
```

```
## Transaction 2 profit: 8.96
```

Here's the profit for the third transaction:

```
transaction.3.profit <-
  (transaction.3.price.per.box * transaction.3.number.of.boxes) - coupon.value

cat(
  "Transaction 3 profit:",
  formatC(
    transaction.3.profit,
    format = "f",
    digits = 2
  )
)
```

```
## Transaction 3 profit: 13.94
```

Here's the profit for the fourth transaction:

```
transaction.4.profit <-
  (transaction.4.price.per.box * transaction.4.number.of.boxes)

cat(
  "Transaction 4 profit:",
  formatC(
    transaction.4.profit,
    format = "f",
    digits = 2
  )
)
```

```
## Transaction 4 profit: 3.99
```

Here's the profit for the fifth transaction:

```
transaction.5.profit <-
  (transaction.5.price.per.box * transaction.5.number.of.boxes) - coupon.value

cat(
  "Transaction 5 profit:",
  formatC(
    transaction.5.profit,
    format = "f",
    digits = 2
  )
)
```

```
## Transaction 5 profit: 16.98
```

```
x <-
  6
```

Here's the profit for the sixth transaction:

```
transaction.6.profit <-
  (transaction.6.price.per.box *
     transaction.6.number.of.boxes) -
     coupon.value

cat(
  "Transaction 6 profit:",
  formatC(
    transaction.6.profit,
    format = "f",
    digits = 2
  )
)
```

```
## Transaction 6 profit: 11.45
```

Finally, we can calculate the total profit:

```
total.profit <-
  transaction.1.profit +
  transaction.2.profit +
  transaction.3.profit +
  transaction.4.profit +
  transaction.5.profit +
  transaction.6.profit

cat(
  "Total profit:",
  formatC(
    x = total.profit,
    format = "f",
    digits = 2
  )
)
```

```
## Total profit: 63.30
```

End of problem 4

Problem 5: VWAP

Suppose a stock has five sales in one day:

- First, 1000 shares are sold at a price of $22.50 per share.
- Next, 200 shares are sold at a price of $24.00 per share.
- Then 750 shares are sold at a price of $23.00 per share.
- Then 800 shares are sold at a price of $24.50 per share.
- Finally, 300 shares are sold at a price of $24.00 per share.

Let's put this into a table:

Transaction	Number of shares	Price per share
1	1000	$22.50
2	200	$24.00
3	750	$23.00
4	800	$24.50
5	300	$24.00

Part (a): Total sales amount

Calculate the total sales amount for this stock. Store this value in a variable, and report it using a `cat()` statement, displaying this value with 2 decimal places.

Solution

```
total.sales.amount <-
    (1000 * 22.50) +
    (200 * 24.00) +
    (750 * 23.00) +
    (800 * 24.50) +
    (300 * 24.00)

cat(
    "Total sales amount:",
    formatC(
        total.sales.amount,
        format = "f",
        digits = 2,
        big.mark = ","
    )
)
```

```
## Total sales amount: 71,350.00
```

Part (b): Total number of shares sold

Calculate the total sales volumne for this stock i.e. the total number of shares sold. Store this value in a variable, and report it using a `cat()` statement.

Solution

```
total.sales.volume <-
    1000 + 200 + 750 + 800 + 300

cat(
    "Total sales volume:",
    total.sales.volume
)
```

```
## Total sales volume: 3050
```

Part (c): Overall average price

Using your results from parts (a) and (b), calculate the volume-weighted average price of this stock. Report your result using a `cat()` statement, displaying this value with 2 decimal places.

Solution

```
overall.average.price <-
    total.sales.amount /
    total.sales.volume

cat(
    "Overall average price:",
    formatC(
        overall.average.price,
        format = "f",
        digits = 2
    )
)
```

```
## Overall average price: 23.39
```

Part (d): Volume-weighted average price

Calculate the volume-weighted average price as a weighted average of the stock prices, where the weights are the relative proportions of the sales volume. Report your result using a `cat()` statement, displaying this value to 2 decimal places.

Solution

```
weight.1 <-
    1000 / total.sales.volume

weight.2 <-
    200 / total.sales.volume
```

```r
weight.3 <-
    750 / total.sales.volume

weight.4 <-
    800 / total.sales.volume

weight.5 <-
    300 / total.sales.volume

vwap <-
    (weight.1 * 22.50) +
    (weight.2 * 24.00) +
    (weight.3 * 23.00) +
    (weight.4 * 24.50) +
    (weight.5 * 24.00)

cat(
    "Volume-weighted average price:",
    formatC(
        vwap,
        format = "f",
        digits = 2
    )
)
```

```
## Volume-weighted average price: 23.39
```

End of problem 5

Problem 6: Profits

A construction firm has 6 projects in three districts:

Project	District	Costs	Revenue
1	Red	277	362
2	Red	315	397
3	Blue	148	159
4	Blue	206	241
5	Green	483	564
6	Green	388	450

The profit for each project is defined as the revenue for the project minus the cost for the project.

Construct a pie chart showing the relative proportions of the total profits for each district.

Solution

First, let's define variables for the cost and revenue for each project:

```
project.1.cost <- 277
project.1.revenue <- 362

project.2.cost <- 315
project.2.revenue <- 397

project.3.cost <- 148
project.3.revenue <- 159

project.4.cost <- 206
project.4.revenue <- 241

project.5.cost <- 483
project.5.revenue <- 564

project.6.cost <- 388
project.6.revenue <- 450
```

Now we can calculate the profit for each project:

```
project.1.profit <-
  project.1.revenue - project.1.cost

project.2.profit <-
  project.2.revenue - project.2.cost

project.3.profit <-
  project.3.revenue - project.3.cost

project.4.profit <-
  project.4.revenue - project.4.cost

project.5.profit <-
```

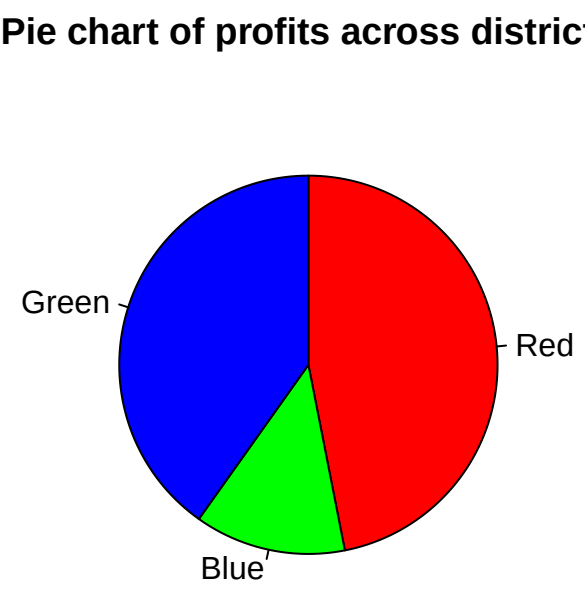
```
  project.5.revenue - project.5.cost

project.6.profit <-
  project.6.revenue - project.6.cost
```

Now we can calculate the profits for each district:

```
red.district.profit <-
  project.1.profit + project.2.profit

blue.district.profit <-
  project.3.profit + project.4.profit

green.district.profit <-
  project.5.profit + project.6.profit
```

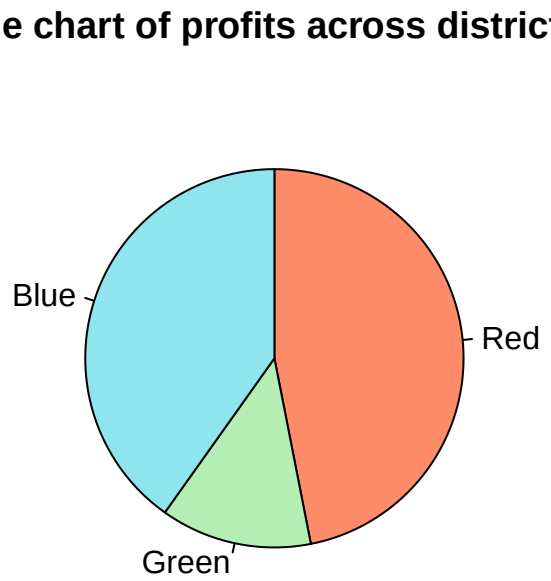
Now we can make the pie chart:

```
pie(
  x = c(red.district.profit, blue.district.profit, green.district.profit),
  labels = c("Red", "Blue", "Green" ),
  main = "Pie chart of profits across districts",
  col = c("red", "green", "blue"),
  clockwise = TRUE
)
```



This is the same pie chart, but with more subdued colors:

```
pie(
  x = c(red.district.profit, blue.district.profit, green.district.profit),
  labels = c("Red", "Green", "Blue"),
  main = "Pie chart of profits across districts",
  col = c("salmon1", "darkseagreen2", "cadetblue2"),
  clockwise = TRUE
)
```



End of Problem 6

Problem 7: The Median Theorem (Extra Credit)

In Euclidean geometry, any two lines that are not parallel will necessarily intersect in one point.

However, in general three lines will not necessarily intersect in a single point, so when that happens it's "special".

There are many theorems in geometry that deal with three lines all intersecting in a point.

One of the simplest of these theorems is called the "Median Theorem".

A *median* is a line segment from the vertex of a triangle to the midpoint of the opposite side.

The Median Theorem states that the three medians of a triangle always intersect in a single point.

In this problem, we'll use R to create a visualization of this theorem by drawing a triangle and then constructing the three medians.

For best results, I suggest this process:

- First, create an empty plot with no data.
- Next, draw in all the line segments.
- After that, draw the points.
- Finally, add the text annotations.

Part (a): Defining the triangle

Let's start by defining the three vertices of a triangle:

Point	x	y
A	2	3
B	4	18
C	9	12

Create variables to store the x- and y-coordinates of these three points.

Try to use some sort of sensible naming conventions for your variables.

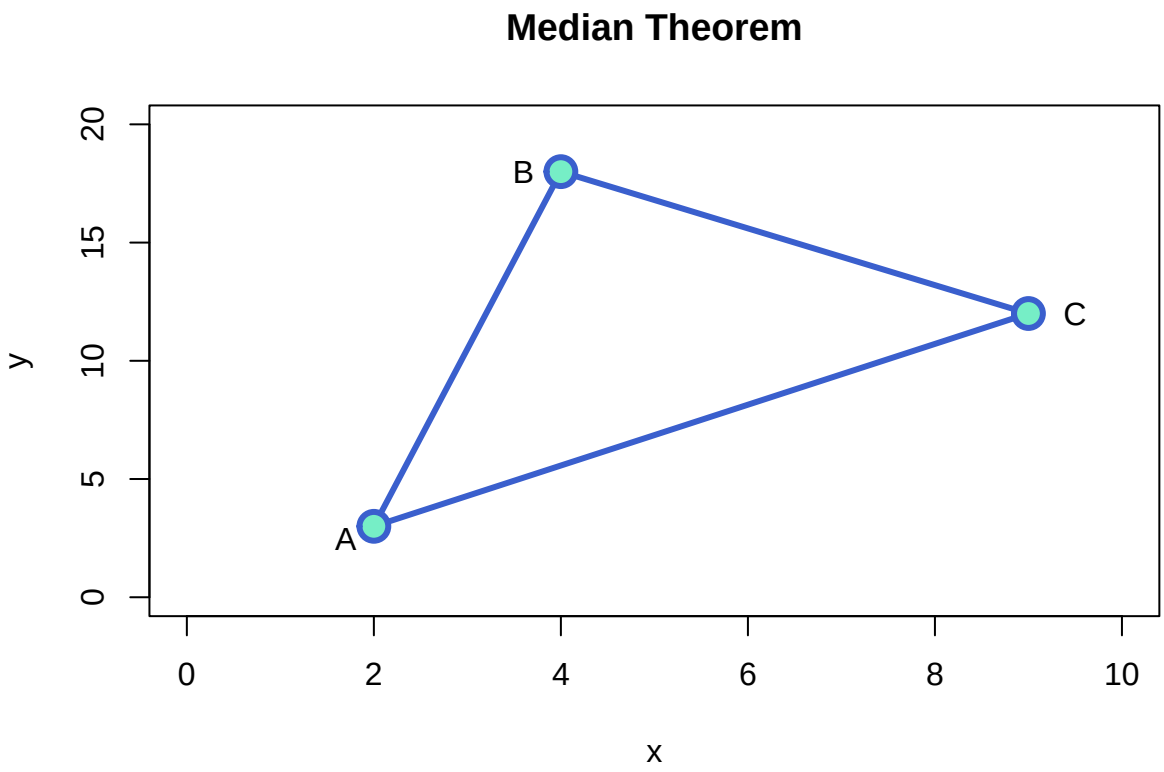
Then create an empty plot with no data, with the x-axis ranging from 0 to 10 and the y-axis ranging from 0 to 20. Be sure to give your graph a main title, and titles for the x- and y-axes.

Finally, draw the triangle ABC. Place points at the vertices, and annotate them with text using the names "A", "B", and "C".

Solution

```
A.x <- 2
A.y <- 3

B.x <- 4
B.y <- 18

C.x <- 9
C.y <- 12
```

```
plot(
  x = NULL,
  xlim = c(0, 10),
  ylim = c(0, 20),
  main = "Median Theorem",
  xlab = "x",
  ylab = "y"
)

x.vector <-
  c(A.x, B.x, C.x)

y.vector <-
  c(A.y, B.y, C.y)

polygon(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  border = "royalblue3"
)

points(
  x = c(A.x, B.x, C.x),
  y = c(A.y, B.y, C.y),
  pch = 21,
  cex = 2,
  lwd = 3,
  col = "royalblue3",
  bg = "aquamarine2"
)

text(
  x = A.x - 0.3,
  y = A.y - 0.5,
  labels = "A"
)

text(
  x = B.x - 0.4,
  y = B.y,
  labels = "B"
)

text(
  x = C.x + 0.5,
  y = C.y,
  labels = "C"
)
```



Part (b): Computing a midpoint

Our next step will be to construct the median from vertex A.

Recall that the median is defined as the line segment from a vertex to the midpoint of the opposite side.

The opposite side from vertex A is the line segment BC, so the median is the line segment from A to the midpoint of BC.

Let's use D to denote the midpoint of the line segment BC, so the median is the line segment AD.

To calculate the x-coordinate of the midpoint of a line segment, take the average of the x-coordinates of the endpoints of the line segment.

That is, take the x-coordinate of the starting point and the x-coordinate of the ending point, add them together, and divide by 2:

$$\text{Midpoint x-coordinate} \;=\; \frac{x0 + x1}{2}$$

Similarly, the y-coordinate of the midpoint of a line segment is the average of the y-coordinates of the endpoints of the line segment:

$$\text{Midpoint y-coordinate} \;=\; \frac{y0 + y1}{2}$$

Calculate the x- and y-coordinates of the point D i.e. the midpoint of the line segment BC. Save your results in variables, and report each using an individual `cat()` statement, displaying the values with 2 decimal places.

Solution

```r
D.x <- (B.x + C.x) / 2

D.y <- (B.y + C.y) / 2
```

```r
cat(
  "D x-coordinate:",
  formatC(
    D.x,
    format = "f",
    digits = 2
  )
)
```

```
## D x-coordinate: 6.50
```

```r
cat(
  "D y-coordinate:",
  formatC(
    D.y,
    format = "f",
    digits = 2
  )
)
```

```
## D y-coordinate: 15.00
```

Part (c): Drawing a median

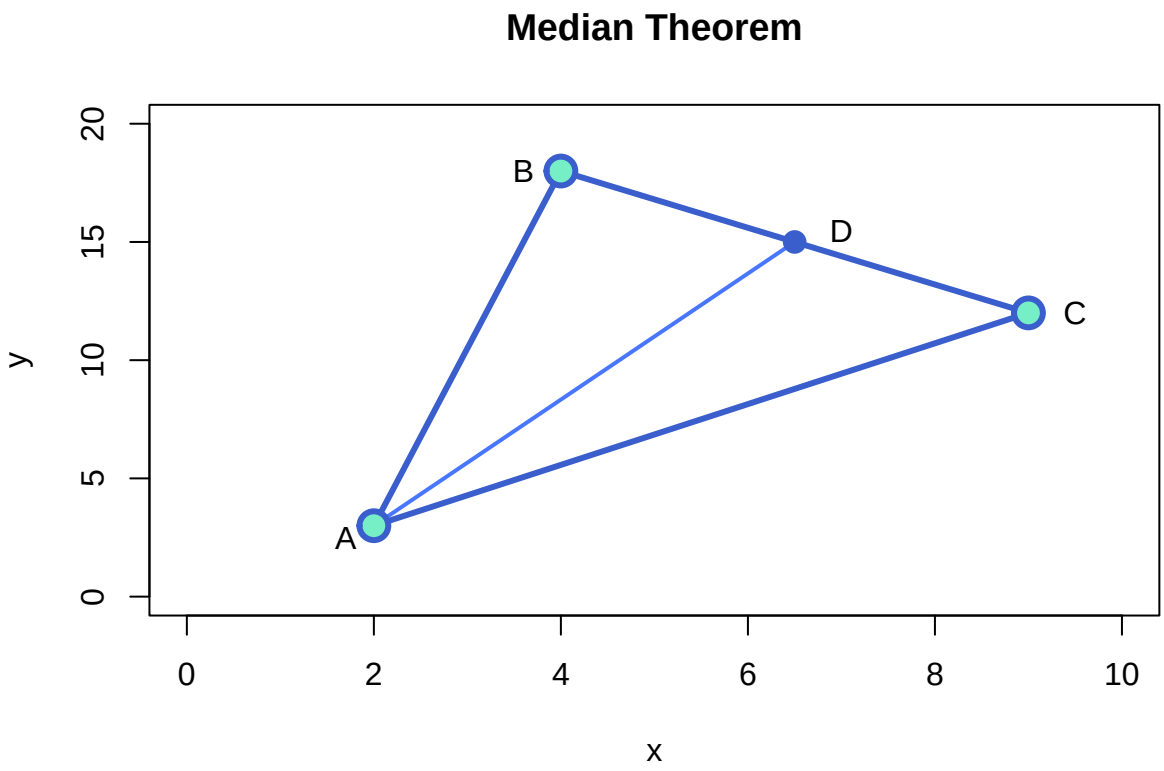
Using your code from part (a), construct a graph of triangle ABC and add in the median line segment from A to the point D that you calculated in part (b). Include a small point at D and a text annotation.

Solution

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

x.vector <-
  c(A.x, B.x, C.x)

y.vector <-
  c(A.y, B.y, C.y)

polygon(
  x = x.vector,
  y = y.vector,
  lty = "solid",
```

```
  lwd = 3,
  border = "royalblue3"
)


# Now we can draw in the median

segments(
  x0 = A.x,
  y0 = A.y,
  x1 = D.x,
  y1 = D.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)

points(
  x = c(A.x, B.x, C.x),
  y = c(A.y, B.y, C.y),
  pch = 21,
  cex = 2,
  lwd = 3,
  col = "royalblue3",
  bg = "aquamarine2"
)

text(
  x = A.x - 0.3,
  y = A.y - 0.5,
  labels = "A"
)

text(
  x = B.x - 0.4,
  y = B.y,
  labels = "B"
)

text(
  x = C.x + 0.5,
  y = C.y,
  labels = "C"
)


# Now let's draw in the median D

points(
  x = D.x,
  y = D.y,
  pch = 19,
  cex = 1.5,
```

```
  col = "royalblue3"
)

text(
  x = D.x + 0.5,
  y = D.y + 0.5,
  labels = "D"
)
```



Part (d): Drawing another median

In this part, we'll draw another median line.

The side opposite vertex B is AC, so the median line segment from B will end at the midpoint of side AC.

Let E denote the midpoint of side AC, so the median line segment is from A to E.

Calculate the x- and y-coordinates of E i.e. the midpoint AC, and store these values in variables.

Then copy over your code from part (c) for drawing the triangle and add code to draw the median line segment BE.

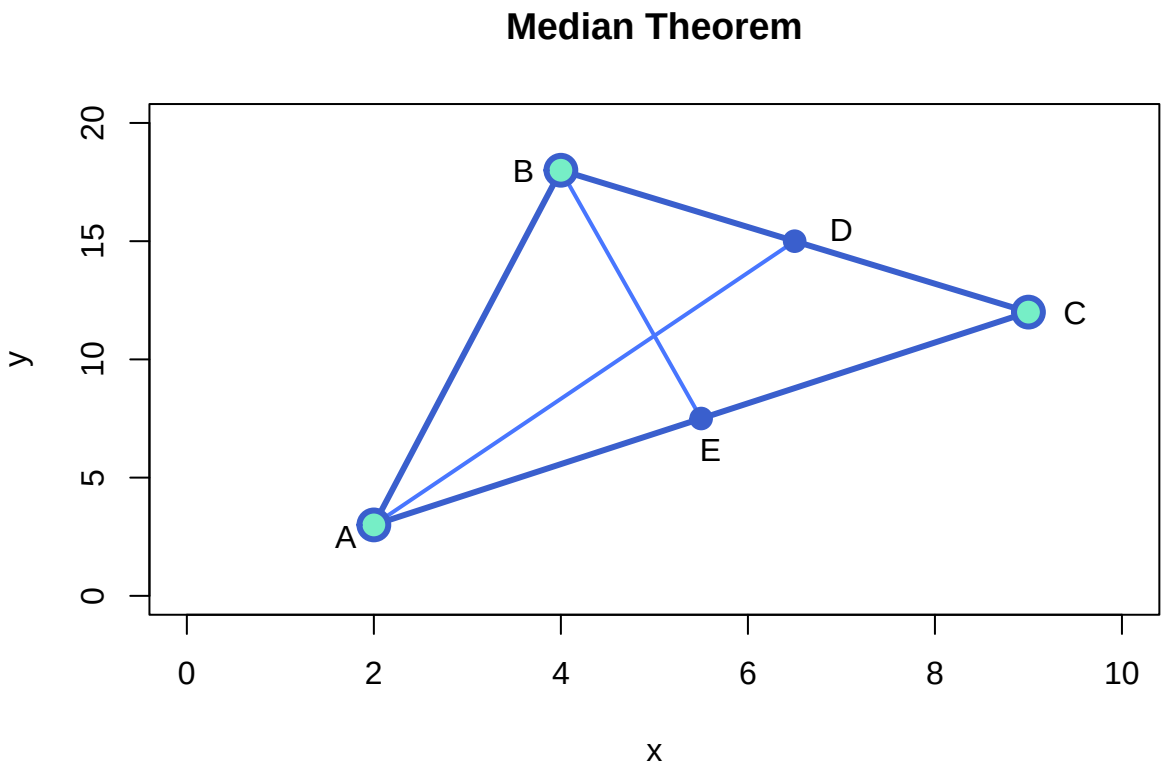
Solution

```
E.x <- (A.x + C.x) / 2

E.y <- (A.y + C.y) / 2
```

```
plot(
  x = NULL,
  xlim = c(0, 10),
  ylim = c(0, 20),
  main = "Median Theorem",
  xlab = "x",
  ylab = "y"
)

x.vector <-
  c(A.x, B.x, C.x)

y.vector <-
  c(A.y, B.y, C.y)

polygon(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  border = "royalblue3"
)


# Now we can draw in the median AD

segments(
  x0 = A.x,
  y0 = A.y,
  x1 = D.x,
  y1 = D.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median BE

segments(
  x0 = B.x,
  y0 = B.y,
  x1 = E.x,
  y1 = E.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)



points(
  x = c(A.x, B.x, C.x),
```

```
  y = c(A.y, B.y, C.y),
  pch = 21,
  cex = 2,
  lwd = 3,
  col = "royalblue3",
  bg = "aquamarine2"
)

text(
  x = A.x - 0.3,
  y = A.y - 0.5,
  labels = "A"
)

text(
  x = B.x - 0.4,
  y = B.y,
  labels = "B"
)

text(
  x = C.x + 0.5,
  y = C.y,
  labels = "C"
)


# Now let's draw in the point D

points(
  x = D.x,
  y = D.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = D.x + 0.5,
  y = D.y + 0.5,
  labels = "D"
)

# Now let's draw in the point E

points(
  x = E.x,
  y = E.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)
```

```
text(
  x = E.x + 0.1,
  y = E.y - 1.3,
  labels = "E"
)
```



Part (e): Drawing the third median

In your graph from part (d), notice that the two medians intersect in a single point.

It would be AMAZING if the third median also went through this point.

In this part, you should copy your code from part (d), and then draw in the third median.

You'll have to figure out what the endpoints are for this third median, and how to calculate them.

When you're all done, the graph should be AMAZING!!

Solution

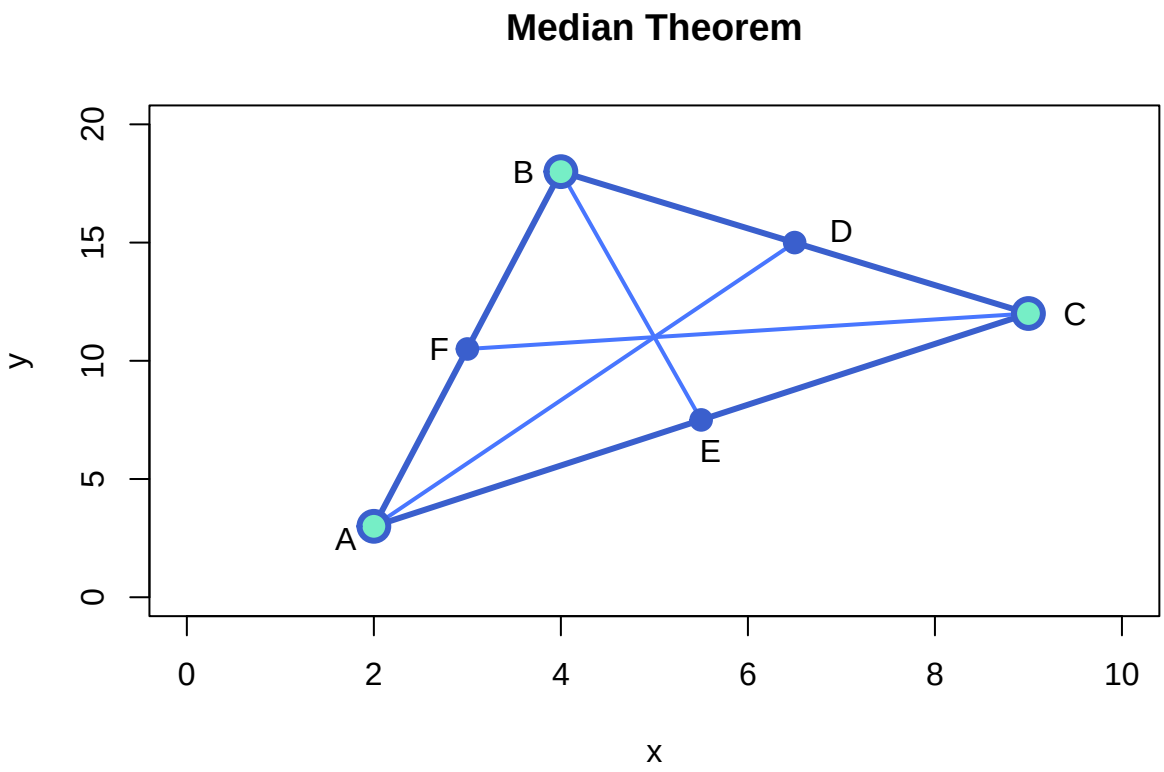
```
F.x <- (A.x + B.x) / 2

F.y <- (A.y + B.y) / 2
```

```
plot(
  x = NULL,
```

```
  xlim = c(0, 10),
  ylim = c(0, 20),
  main = "Median Theorem",
  xlab = "x",
  ylab = "y"
)

x.vector <-
  c(A.x, B.x, C.x)

y.vector <-
  c(A.y, B.y, C.y)

polygon(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  border = "royalblue3"
)


# Now we can draw in the median AD

segments(
  x0 = A.x,
  y0 = A.y,
  x1 = D.x,
  y1 = D.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median BE

segments(
  x0 = B.x,
  y0 = B.y,
  x1 = E.x,
  y1 = E.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median CF

segments(
  x0 = C.x,
  y0 = C.y,
```

```
  x1 = F.x,
  y1 = F.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)

points(
  x = c(A.x, B.x, C.x),
  y = c(A.y, B.y, C.y),
  pch = 21,
  cex = 2,
  lwd = 3,
  col = "royalblue3",
  bg = "aquamarine2"
)

text(
  x = A.x - 0.3,
  y = A.y - 0.5,
  labels = "A"
)

text(
  x = B.x - 0.4,
  y = B.y,
  labels = "B"
)

text(
  x = C.x + 0.5,
  y = C.y,
  labels = "C"
)


# Now let's draw in the point D

points(
  x = D.x,
  y = D.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = D.x + 0.5,
  y = D.y + 0.5,
  labels = "D"
)

# Now let's draw in the point E
```

```
points(
  x = E.x,
  y = E.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = E.x + 0.1,
  y = E.y - 1.3,
  labels = "E"
)


# Now let's draw in the point E

points(
  x = F.x,
  y = F.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = F.x - 0.3,
  y = F.y,
  labels = "F"
)
```



Part (f): Another triangle

Let's try this again with another triangle:

Point	x	y
A	7	4
B	2	15
C	8	10

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. Include a main title and titles for the x- and y-axes.

Draw the triangle ABC along with the three medians.

You are welcome and encouraged to copy your code from above and modify it.

If you wrote your code properly, this problem should be very easy, because all you have to do is to alter the values for the coordinates of the three points and then everything else should be defined in terms of these three variables.

Most of your work will be in adjusting the position of the text labels so they look good with these points.

Solution

```
A.x <- 7
A.y <- 4

B.x <- 2
B.y <- 15

C.x <- 8
C.y <- 10
```

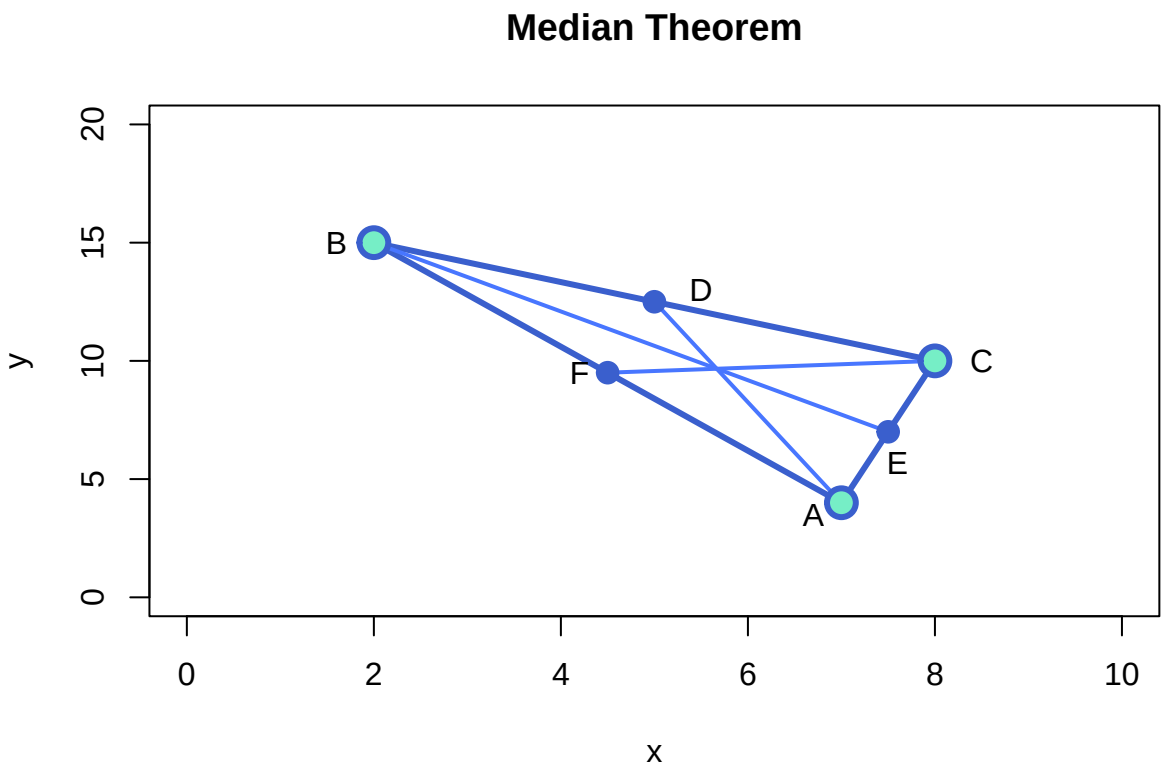
```
D.x <- (B.x + C.x) / 2

D.y <- (B.y + C.y) / 2

E.x <- (A.x + C.x) / 2

E.y <- (A.y + C.y) / 2

F.x <- (A.x + B.x) / 2

F.y <- (A.y + B.y) / 2
```

```
plot(
  x = NULL,
  xlim = c(0, 10),
  ylim = c(0, 20),
  main = "Median Theorem",
  xlab = "x",
  ylab = "y"
)

x.vector <-
  c(A.x, B.x, C.x)

y.vector <-
  c(A.y, B.y, C.y)

polygon(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  border = "royalblue3"
)


# Now we can draw in the median AD

segments(
  x0 = A.x,
  y0 = A.y,
  x1 = D.x,
  y1 = D.y,
  lty = "solid",
```

```
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median BE

segments(
  x0 = B.x,
  y0 = B.y,
  x1 = E.x,
  y1 = E.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median CF

segments(
  x0 = C.x,
  y0 = C.y,
  x1 = F.x,
  y1 = F.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)

points(
  x = c(A.x, B.x, C.x),
  y = c(A.y, B.y, C.y),
  pch = 21,
  cex = 2,
  lwd = 3,
  col = "royalblue3",
  bg = "aquamarine2"
)

text(
  x = A.x - 0.3,
  y = A.y - 0.5,
  labels = "A"
)

text(
  x = B.x - 0.4,
  y = B.y,
  labels = "B"
)

text(
```

```
  x = C.x + 0.5,
  y = C.y,
  labels = "C"
)


# Now let's draw in the point D

points(
  x = D.x,
  y = D.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = D.x + 0.5,
  y = D.y + 0.5,
  labels = "D"
)

# Now let's draw in the point E

points(
  x = E.x,
  y = E.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = E.x + 0.1,
  y = E.y - 1.3,
  labels = "E"
)


# Now let's draw in the point E

points(
  x = F.x,
  y = F.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)

text(
  x = F.x - 0.3,
  y = F.y,
  labels = "F"
```

```
)
```



Part (g): Randomized example generator

You might be wondering if I picked a special triangle where all the three medians just happen to intersect.

Is the Median Theorem really true for ALL triangles?

The only way to definitively answer this question is by a formal mathematical proof (the answer is yes, the theorem is always true).

But another approach to this question is to look at a lot of randomly generated triangles, and if the theorem is true for a sequence of these randomly generated examples then that's a strong indication that it's true in general.

We can use the `runif()` function to generate random values on an interval, which takes three input arguments:

- The number of random values generated, denoted `n`.
- The minimum of the interval, denoted `min`.
- The maximum of the interval, denoted `max`.

To generate one random value on the interval from 0 to 10, we have:

```
runif(
  n = 1,
  min = 0,
  max = 10
)
```

```
## [1] 4.657292
```

In our example, the x-axis ranged from 0 to 10, so we can generate a random value for the x-coordinate of the point A:

```
A.x <-
  runif(
    n = 1,
    min = 0,
    max = 10
  )
```

Similarly, the y-axis ranged from 0 to 20, so we can generate a random value for the y-coordinate of the point A:

```
A.y <-
  runif(
    n = 1,
    min = 0,
    max = 20
  )
```

In this problem, use the `runif()` function to generate random values for the x- and y-coordinates of the points A, B, and C.

Then draw the triangle ABC and add the medians.

If you do all of this in a single code chunk, then each time you run the code chunk you'll get a different random triangle, and you should see that the Median Theorem is true in every case.

Hint: if you wrote your code properly in the previous examples, this should be very easy to implement, because once you've generated the random values for A, B, and C then everything else should be defined in terms of these variables so you can just copy your code from above with little modification.

Don't worry about text annotations; it's hard to get the placement right when the points are randomly generated, and anyway all we're doing here is looking to see if the three medians intersect.

Solution

```
A.x <-
    runif(
        n = 1,
        min = 0,
        max = 10
    )

A.y <- runif(n = 1, min = 0, max = 20)

B.x <- runif(n = 1, min = 0, max = 10)
```

```r
B.y <- runif(n = 1, min = 0, max = 20)

C.x <- runif(n = 1, min = 0, max = 10)
C.y <- runif(n = 1, min = 0, max = 20)


D.x <- (B.x + C.x) / 2
D.y <- (B.y + C.y) / 2

E.x <- (A.x + C.x) / 2
E.y <- (A.y + C.y) / 2

F.x <- (A.x + B.x) / 2
F.y <- (A.y + B.y) / 2

plot(
  x = NULL,
  xlim = c(0, 10),
  ylim = c(0, 20),
  main = "Median Theorem",
  xlab = "x",
  ylab = "y"
)

x.vector <-
  c(A.x, B.x, C.x)

y.vector <-
  c(A.y, B.y, C.y)

polygon(
  x = x.vector,
  y = y.vector,
  lty = "solid",
  lwd = 3,
  border = "royalblue3"
)


# Now we can draw in the median AD

segments(
  x0 = A.x,
  y0 = A.y,
  x1 = D.x,
  y1 = D.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median BE
```

```
segments(
  x0 = B.x,
  y0 = B.y,
  x1 = E.x,
  y1 = E.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)


# Now we can draw in the median CF

segments(
  x0 = C.x,
  y0 = C.y,
  x1 = F.x,
  y1 = F.y,
  lty = "solid",
  lwd = 2,
  col = "royalblue1"
)

points(
  x = c(A.x, B.x, C.x),
  y = c(A.y, B.y, C.y),
  pch = 21,
  cex = 2,
  lwd = 3,
  col = "royalblue3",
  bg = "aquamarine2"
)



# Now let's draw in the point D

points(
  x = D.x,
  y = D.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)



# Now let's draw in the point E

points(
  x = E.x,
  y = E.y,
  pch = 19,
```

```
  cex = 1.5,
  col = "royalblue3"
)


# Now let's draw in the point F

points(
  x = F.x,
  y = F.y,
  pch = 19,
  cex = 1.5,
  col = "royalblue3"
)
```