Problem Set 11 Solutions

MATH E-156: Mathematical Statistics

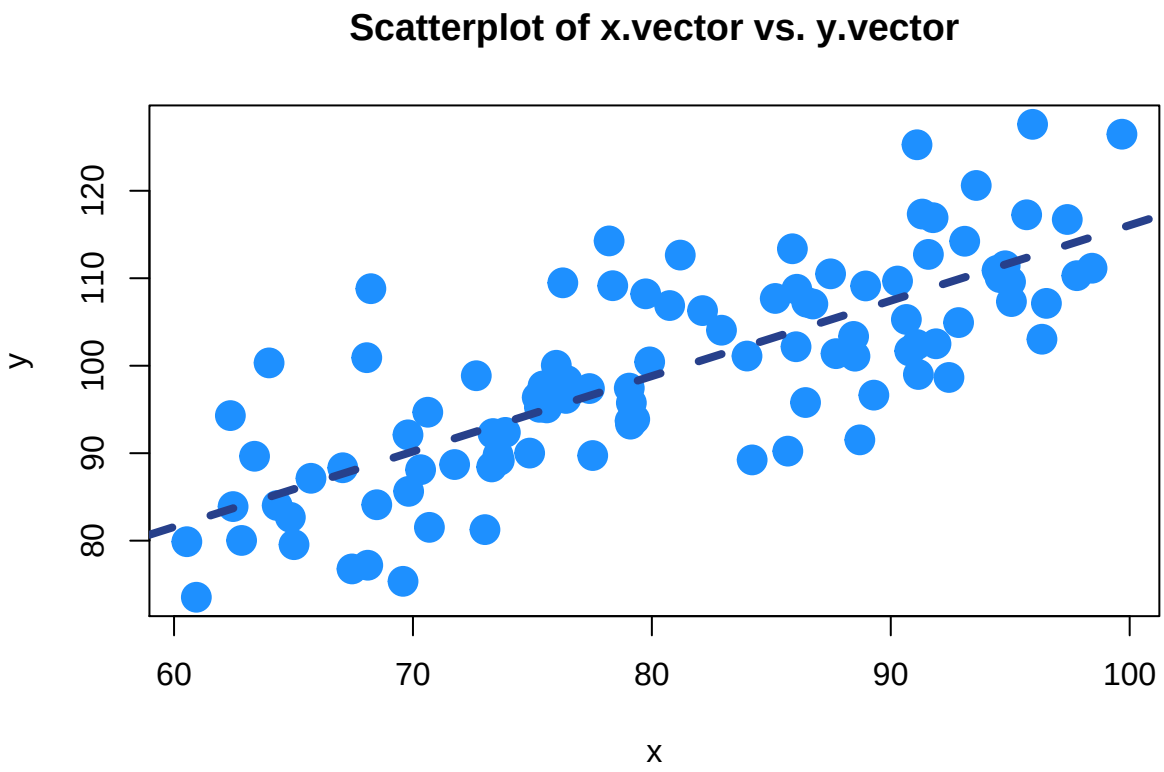
```
load( "Problem Set 11 R Objects.Rdata" )
```

Problem 1: Scatterplot

Construct a scatterplot of the values in `problem.1.x.vector` and `problem.1.y.vector`. Then fit a least-squares regression line to this scatterplot. (The term "least-squares" means that the y-intercept and slope parameters for the line are estimated by minimizing the sum of squared errors.)

Solution

```
plot(
    problem.1.x.vector,
    problem.1.y.vector,
    main = "Scatterplot of x.vector vs. y.vector",
    xlab = "x",
    ylab = "y",
    pch = 19,
    cex = 2,
    col = "dodgerblue1"
)

linear.model <-
    lm( problem.1.y.vector ~ problem.1.x.vector )

abline(
    linear.model,
    lty = "dashed",
    lwd = 4,
    col = "royalblue4"
)
```


Scatterplot of x.vector vs. y.vector
y
80
90
100
110
120
60
70
80
90
100
x

End of problem 1

Problem 2: Independence of MST and MSE

In lecture, we saw that we could investigate the independence of the sample mean and the sample variance for a normal sample by generating a random sample of sample means and sample variances and then creating a scatterplot.

I mentioned that we use this idea about the independence of the sample mean and the sample variance in other situations, notably when constructing the F statistic in ANOVA.

In this problem, we will perform a similar simulation to study this idea in the context of ANOVA.

- First, create two outcome vectors, one for the MST and one for the MSE.
- Use 1,000 simulation replications (that's not a typo).
- For each simulation replication:
  - Generate 3 independent normal random samples, each with an expected value of 80, a standard deviation of 5, and a sample size of $n = 20$.
  - Calculate the sample means of each group, and store them in a vector.
  - Calculate the sample variances of each group, and store them in a vector.
  - Calculate the MST as the sample variance of the vector of sample means, times the number of groups (i.e. 3).
  - Calculate the MSE as the sample mean of the vector of sample variances.
  - Store the MST in the MST outcome vector, and the MSE in the MSE outcome vector.

When the simulation is done, the MST outcome vector will be populated with random values of the MST under the null hypothesis, and the MSE outcome vector will be populated with random values of the MSE under the null hypothesis.

There's nothing to report here, but write your code clearly so the TAs can understand what you're doing.

Solution

```
sample.size <- 20

number.of.groups <- 3


number.of.replications <- 1000

mst.outcome.vector <- numeric( number.of.replications )

mse.outcome.vector <- numeric( number.of.replications )

for( replication.index in 1:number.of.replications ) {

    group.a.random.sample <-
        rnorm( sample.size, mean = 80, sd = 5)

    group.b.random.sample <-
        rnorm( sample.size, mean = 80, sd = 5)

    group.c.random.sample <-
```

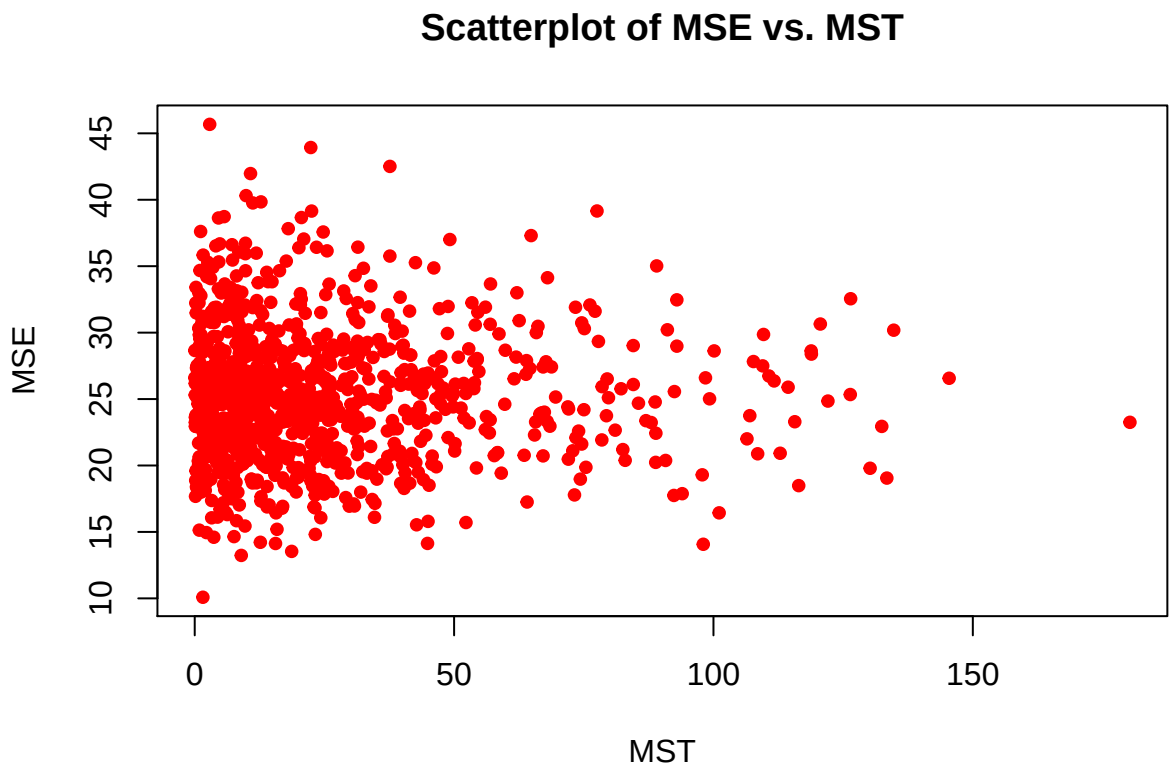
```
        rnorm( sample.size, mean = 80, sd = 5)

    group.a.sample.mean <-
        mean( group.a.random.sample )

    group.b.sample.mean <-
        mean( group.b.random.sample )

    group.c.sample.mean <-
        mean( group.c.random.sample )

    group.mean.vector <-
        c(
            group.a.sample.mean,
            group.b.sample.mean,
            group.c.sample.mean
        )

    group.a.sample.variance <-
        var( group.a.random.sample )

    group.b.sample.variance <-
        var( group.b.random.sample )

    group.c.sample.variance <-
        var( group.c.random.sample )

    group.variance.vector <-
        c(
            group.a.sample.variance,
            group.b.sample.variance,
            group.c.sample.variance
        )

    mean.square.for.treatments <-
        sample.size * var( group.mean.vector )

    mean.square.for.errors <-
        mean( group.variance.vector )

    mst.outcome.vector[ replication.index ] <-
        mean.square.for.treatments

    mse.outcome.vector[ replication.index ] <-
        mean.square.for.errors
}
```

Part (b)

Make a scatterplot of the MSE outcome vector versus the MST outcome vector.

Solution

```r
plot(
    mst.outcome.vector,
    mse.outcome.vector,
    main = "Scatterplot of MSE vs. MST",
    xlab = "MST",
    ylab = "MSE",
    pch = 19,
    cex = 0.8,
    col = "red"
)
```



Part (c)

Using your graph in part (b), what do you conclude about the independence of the MST and the MSE, under the null hypothesis? Explain your answer with a few sentences.

Solution

By visual inspection, the distribution of the values for MSE does not seem to vary across the range of the MST. This indicates that the MST and MSE are independent.

End of problem 2

Problem 3: Visualizing the Simple Linear Regression Model

Let's consider a simple linear regression model:

- The experiment will use 5 different levels for the x variable: 60, 70, 80, 90, and 100.
- For each value of x, there are 8 observations.
- The y-intercept parameter for this model is $\alpha = 40$.
- The slope parameter for this model is $\beta = 1.2$.
- The variance of the noise is $\sigma^2 = 80$.

Our goal in this problem is to construct a visualization of this model by generating a random sample according to the parameters and then displaying the values using a scatterplot.

Part (a)

In this part, we will generate the random sample.

- First, create a vector consisting of all the x values.
- Second, define variables for the y-intercept parameter α, the slope parameter β, and the noise variance σ^2.
- Next, construct a vector of the noise values for the experiment.
- Use vectorized operations to construct a vector of y values.

Remember that while there are 5 different levels for x, there are 8 observations per level, so there are a total of 40 observations in the entire sample. That means that the vector of x values, the value of noise values, and the vector of y values will all consist of 40 elements.

Here are some variables for you:

```
number.of.levels <- 5

observations.per.level <- 8
```

Solution

```
x.vector <-
    c(
        rep( 60, times = observations.per.level),
        rep( 70, times = observations.per.level),
        rep( 80, times = observations.per.level),
        rep( 90, times = observations.per.level),
        rep( 100, times = observations.per.level)
    )

y.intercept.parameter <- 40

slope.parameter <- 1.2
```

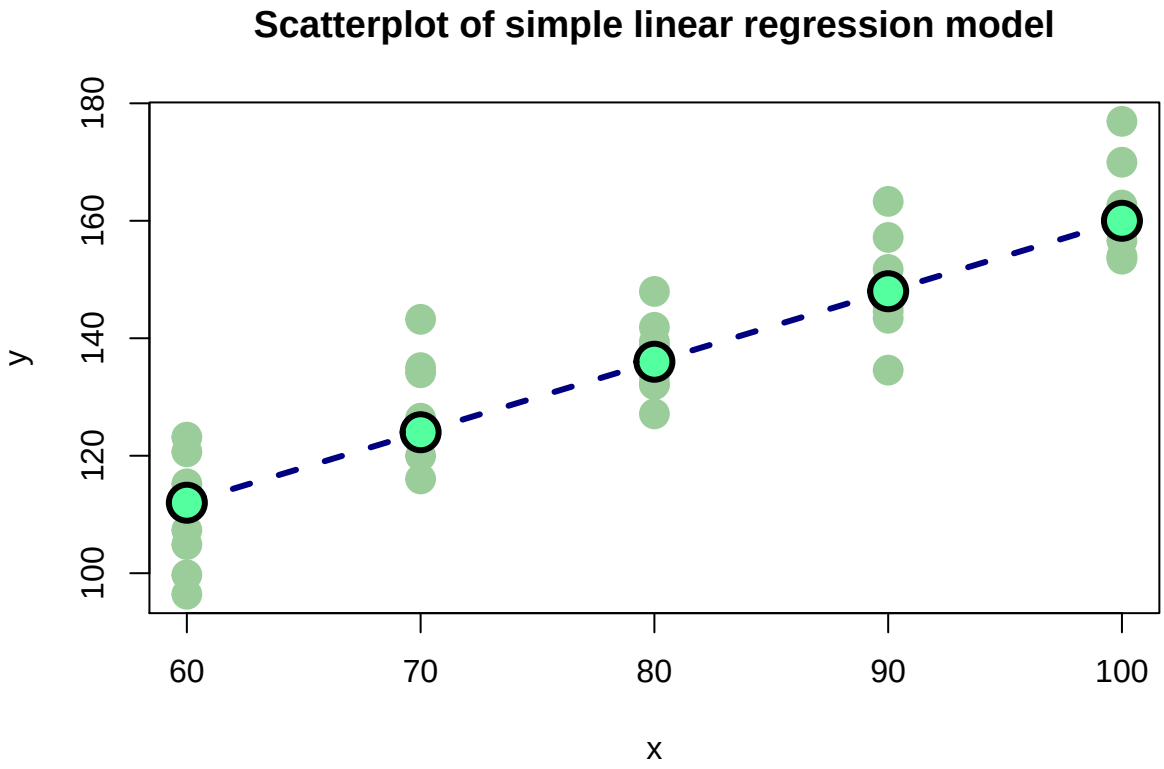
```
noise.variance <- 80

noise.vector <-
    rnorm(
        number.of.levels * observations.per.level,
        mean = 0,
        sd = sqrt( noise.variance )
    )

y.vector <-
    slope.parameter * x.vector +
    y.intercept.parameter +
    noise.vector
```

Part (b)

Create a scatterplot of the values in the x vector and the y vector. Include a line showing the expected value of Y as a linear function of x.

Solution

```
plot(
    x.vector,
    y.vector,
    main = "Scatterplot of simple linear regression model",
    xlab = "x",
    ylab = "y",
    pch = 19,
    cex = 2,
    col = "darkseagreen3"
)

segments(
    x0 = 60,
    y0 = slope.parameter * 60 + y.intercept.parameter,
    x1 = 100,
    y1 = slope.parameter * 100 + y.intercept.parameter,
    lty = "dashed",
    lwd = 3,
    col = "navy"
)

x.levels.vector <-
    c( 60, 70, 80, 90, 100 )

points(
    x.levels.vector,
    slope.parameter * x.levels.vector + y.intercept.parameter,
    pch = 21,
    lwd = 3,
    cex = 2.5,
    bg = "seagreen1"
)
```


Scatterplot of simple linear regression model
y
100
120
140
160
180
60
70
80
90
100
x

End of problem 3

Problem 4

The vector `problem.4.data` contains a set of random values.

Part (a)

Calculate the sample mean of the values in `problem.4.data`. Report your result using a `cat()` statement, rounding to 5 decimal places.

Solution

```
sample.mean <-
    mean( problem.4.data )

cat( "Sample mean:",
     round( sample.mean, 5 ) )
```

```
## Sample mean: 25.3861
```

Part (b)

Construct a sequence of values starting at 24 and going up to 27 in steps of 0.1. Display this sequence using a `cat()` statement.

Solution

```
sequence.vector <-
    seq( from = 24, to = 27, by = 0.1 )

cat( "Sequence vector:", sequence.vector )
```

```
## Sequence vector: 24 24.1 24.2 24.3 24.4 24.5 24.6 24.7 24.8 24.9 25 25.1 25.2 25.3 25.4 25.5 25.6 25
```

Part (c)

For each value in the sequence you constructed in part (b), calculate the sum of the squared errors with the values in `problem.4.data`. Store these values in a vector, and display this vector using a `cat()` statment, rounding to 5 decimal places.

Solution

```
sequence.length <-
    length( sequence.vector )

squared.error.vector <-
    numeric( sequence.length )

for( sequence.index in 1:sequence.length ) {

    current.element <-
        sequence.vector[ sequence.index ]
```

```
    sum.of.squared.errors <-
        sum( (problem.4.data - current.element)^2 )

    squared.error.vector[ sequence.index ] <-
        sum.of.squared.errors
}

cat( "Sum of squared errors:",
     round( squared.error.vector, 5 ) )
```

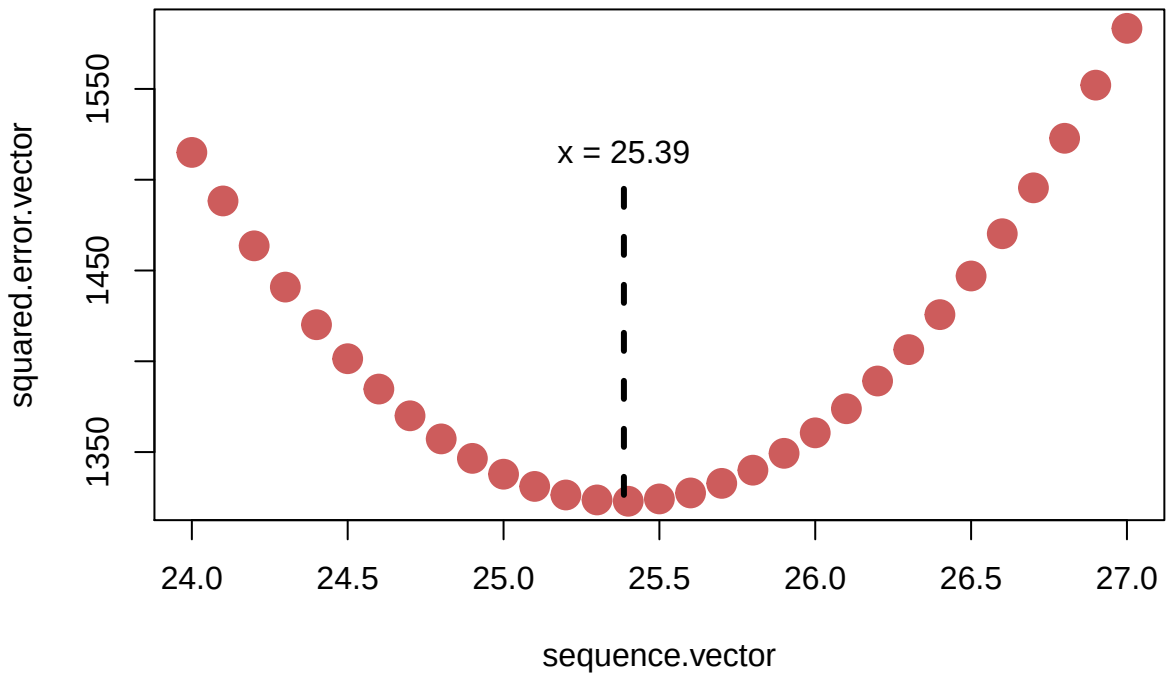
```
## Sum of squared errors: 1515.027 1488.305 1463.583 1440.861 1420.139 1401.417 1384.695 1369.973 1357.
```

Part (d)

Create a scatterplot of the vector of squared errors from part (c) versus the sequence from part (b). Then draw a vertical line indicating the sample mean you calculated in part (a) and annotate it with text.

Solution

```
plot(
    sequence.vector,
    squared.error.vector,
    pch = 19,
    cex = 2,
    col = "indianred"
)

segments(
    x0 = sample.mean,
    y0 = 1300,
    x1 = sample.mean,
    y1 = 1500,
    lty = "dashed",
    lwd = 3,
    col = "black"
)

text(
    x = sample.mean,
    y = 1515,
    labels =
        paste( "x =", round( sample.mean, 2 ) )
)
```


squared.error.vector
1350
1450
1550
x = 25.39
24.0
24.5
25.0
25.5
26.0
26.5
27.0
sequence.vector

End of problem 4

Problem 5

The vectors `problem.5.x.vector` and `problem.5.y.vector` contain a random sample of values from the linear regression model that we studied in problem 3.

Part (a)

Calculate the sample mean of the values in `problem.5.x.vector`. Store your result in a variable, and report it using a `cat()` statement, rounding to 5 decimal places.

Solution

```
x.sample.mean <-
    mean( problem.5.x.vector )

cat( "X sample mean:",
     round( x.sample.mean, 5 ) )
```

```
## X sample mean: 80
```

Part (b)

Calculate the sample mean of the values in `problem.5.y.vector`. Store your result in a variable, and report it using a `cat()` statement, rounding to 5 decimal places.

Solution

```
y.sample.mean <-
    mean( problem.5.y.vector )

cat( "Y sample mean:",
     round( y.sample.mean, 5 ) )
```

```
## Y sample mean: 136.6067
```

Part (c)

Calculate S_{xy} for this data. Report your result using a `cat()` statement, rounding to 5 decimal places.

```
s.xy <-
    sum(
        (problem.5.x.vector - x.sample.mean) *
            (problem.5.y.vector - y.sample.mean)
    )

cat( "S_xy =",
     round( s.xy, 5 ) )
```

```
## S_xy = 10378.86
```

Part (d)

Calculate S_{xx} for this data. Report your result using a `cat()` statement, rounding to 5 decimal places.

```
s.xx <-
    sum( (problem.5.x.vector - x.sample.mean)^2 )

cat( "S_xx =",
     round( s.xx, 5 ) )
```

```
## S_xx = 8000
```

Part (e)

Calculate $\widehat{\beta}$, the estimate of the slope parameter. Report your result using a `cat()` statement.

Solution

```
slope.parameter.estimate <-
    s.xy / s.xx

cat( "Slope parameter estimate:",
     round( slope.parameter.estimate, 5 ) )
```

```
## Slope parameter estimate: 1.29736
```

Part (f)

Calculate $\widehat{\alpha}$, the estimate of the y-intercept parameter. Report your result using a `cat()` statement.

Solution

```
y.intercept.parameter.estimate <-
    y.sample.mean -
    slope.parameter.estimate * x.sample.mean

cat( "y-intercept parameter estimate:",
     round( y.intercept.parameter.estimate, 5 ) )
```

```
## y-intercept parameter estimate: 32.81806
```

Part (g)

Use the built-in R function `lm()` to estimate the linear model. Then report the y-intercept parameter estimate and the slope parameter estimate using a separate `cat()` statement for each, rounding to 5 decimal places. How do these values compare with your results in parts (e) and (f)?

Solution

```
linear.model <-
    lm( problem.5.y.vector ~ problem.5.x.vector )
```

```
y.intercept.parameter.estimate <-
    linear.model$coefficients[ "(Intercept)" ]

cat( "y-intercept parameter estimate:",
     round( y.intercept.parameter.estimate, 5 ) )
```

```
## y-intercept parameter estimate: 32.81806
```

```
slope.parameter.estimate <-
    linear.model$coefficients[ "problem.5.x.vector" ]

cat( "Slope parameter estimate:",
     round( slope.parameter.estimate, 5 ) )
```

```
## Slope parameter estimate: 1.29736
```

End of problem 5

Problem 6

This problem is a continuation of problem 5.

We saw in lecture that the slope parameter estimate $\widehat{\beta}$ was calculated by minimizing the sum of squared errors.

In this problem we will visualize this concept.

Suppose we have a value a for the y-intercept parameter and a value b for the slope parameter.Then we can then construct the *fitted values* for y using this formula:

$$\widehat{y_i} \;=\; b \cdot x_i + a$$

In other words, to calculate the fitted value of y_i we just calculate the linear formula given x_i, using the values a and b for the y-intercept and slope.

Notice that the way that I've described these fitted values we are not requried to use the least-squares parameter estimates $\widehat{\alpha}$ and $\widehat{\beta}$ that we've calculated in problem 5.

Instead, a and b can be anything, although of course they *could* be the parameter estimates $\widehat{\alpha}$ and $\widehat{\beta}$.

Then the sum of the squared errors for a and b is the sum of the squared differences of the observed values y_i minus the fitted values $\widehat{y}_i$:

$$\text{Sum of squared errors} \;=\; \sum_{i=1}^{n}(y_i - \widehat{y}_i)^2$$

We can also write this as:

$$\text{Sum of squared errors} \;=\; \sum_{i=1}^{n}(y_i - (b \cdot x_i + a))^2$$

Our strategy for this problem will be to construct a range of values for the slope coefficient, and then to calculate the sum of the squared errors for each value in this range.

For all of these calculations we'll still use the y-intercept parameter estimate that we found in problem 5 for the value of a.

For instance, suppose that the value of b is 0.2.

Then the sum of the squared errors would be:

$$\text{Sum of squared errors} \;=\; \sum_{i=1}^{n}(y_i - 0.2 \cdot x_i + \widehat{\alpha}))^2$$

Finally, we'll create a scatterplot of these sums of squared errors, and you should see that the minimum value of these squared errors occurs at the actual slope parameter estimate $\widehat{\beta}$ that we calculated in problem 5.

Part (a)

Construct a sequence of values starting at 0 and going up to 2, in increments of 0.1. Report your result using a `cat()` statement.

Solution

```
sequence.vector <-
    seq( from = 0, to = 2, by = 0.1 )

cat( "Sequence vector:",
     sequence.vector )
```

```
## Sequence vector: 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2
```

Part (b)

Now we will calculate the sums of squared errors for each value of the sequence from part (a).

- First, construct a storage vector to hold the sum of squared errors for each value of the sequence.
- Next, for each value b in the sequence, calculate the sum of the squared errors between the observed values y_i and the fitted values, using the value b for the slope and the y-intercept parameter from problem 5.
- Store each sum of squared errors in the storage vector.

There's nothing to report here, but write your code clearly so that the TAs can understand what you're doing.

Hint: this is similar (but not indentical!) to what you did in problem 4, part (c), so you can look there for inspiration.

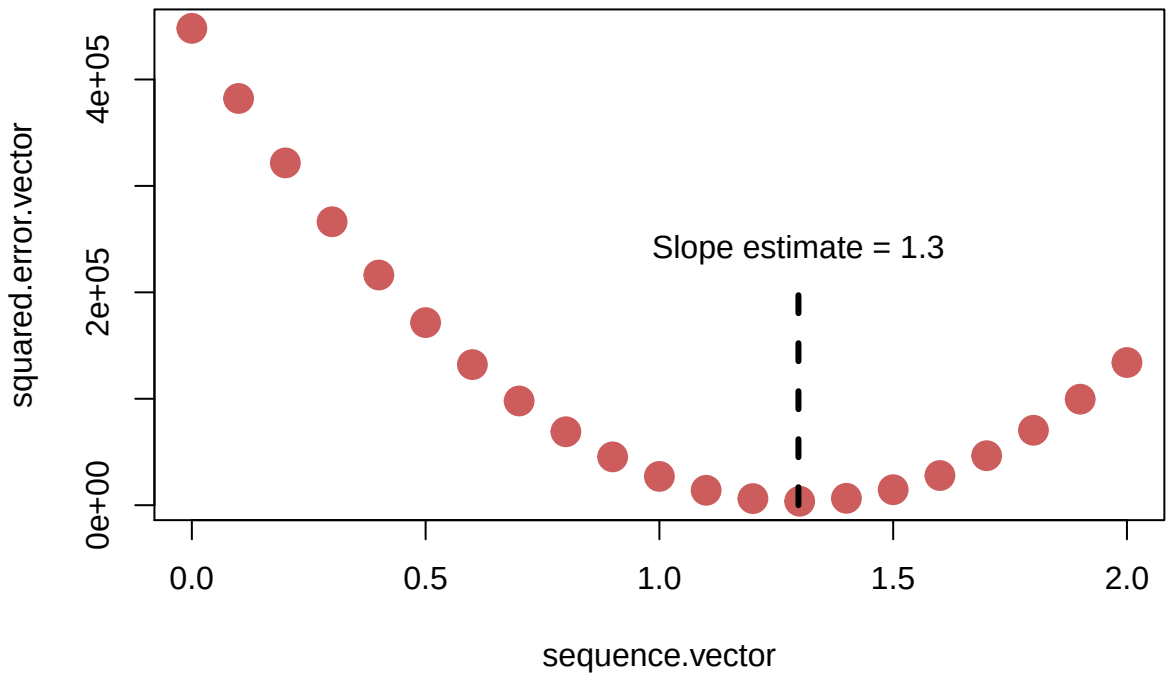
Solution

```
sequence.length <-
    length( sequence.vector )

squared.error.vector <-
    numeric( sequence.length )

for( sequence.index in 1:sequence.length ) {

    current.element <-
        sequence.vector[ sequence.index ]

    sum.of.squared.errors <-
        sum( (problem.5.y.vector -
                  (current.element * problem.5.x.vector +
                       y.intercept.parameter.estimate) )^2 )

    squared.error.vector[ sequence.index ] <-
        sum.of.squared.errors
}
```

Part (c)

Draw a scatterplot of the values of sums of the squared errors versus the sequence of values for the slope. Then draw a vertical line indicating the slope parameter from problem 5, and annotate this with text.

```
plot(
    sequence.vector,
    squared.error.vector,
    pch = 19,
    cex = 2,
    col = "indianred"
)

segments(
    x0 = slope.parameter.estimate,
    y0 = 0,
    x1 = slope.parameter.estimate,
    y1 = 220000,
    lty = "dashed",
    lwd = 3,
    col = "black"
)

text(
    x = slope.parameter.estimate,
    y = 240000,
    labels =
        paste(
            "Slope estimate =",
            round( slope.parameter.estimate, 2 ) )
)
```


squared.error.vector
0e+00
2e+05
4e+05
Slope estimate = 1.3
0.0
0.5
1.0
1.5
2.0
sequence.vector