Notes for the structure of this coursework.

Some questions may involve R codes. I first talk about the methods I used, outlines of the program implementation, result (like plots and calculate values), and explanations.

And at the end of the part of the question if I have used R code, You will see these lines

################## R Script ##################

…

…. R code for that part of the question …

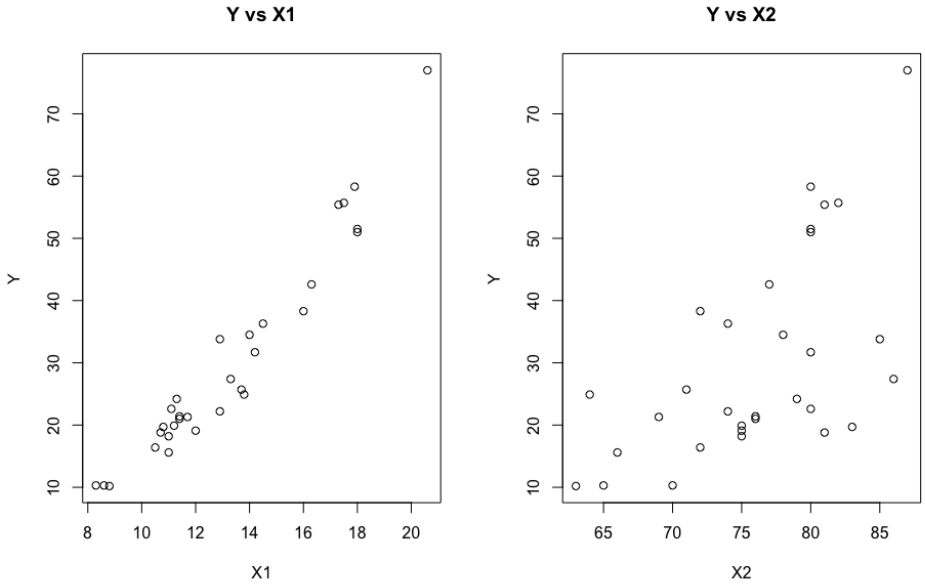
...

##

Q1

a) Read the data file into R and investigate whether or not there is any graphical evidence for relationships between Y and the predictor variables.

Please also see the r script below to know how I plot the scatter plot



From the plot of Y against each predictor variable,

Y against X1: it is very clear that Y and X1 are positively correlated and looks like a linear trend.

Y against X2: this is also clear that Y and X2 are positively correlated, as we can see, if X2 is getting larger, Y is getting larger as well. However, this is not obvious if there is a linear trend.

R Script Q1a##################

Q1 a)

change the csv file location

data = read.csv("~/Documents/uon/MATH4065/assessment/ExperimentData.csv")

par(mfrow=c(1,2))

plot(Y~X1, data = data, main="Y vs X1")

plot(Y~X2, data = data, main="Y vs X2")

par(mfrow=c(1,1))

##

b) Using R, fit a linear model (model 1) in which Y is the response variable, $X1$ is the predictor variable and there is also an intercept term. Find the least-squares estimates of the model parameters. Is there evidence that $X1$ can explain the variation in Y, or not?

please also see the R script for the use of R to fit a linear model

Outlines:

i) We would like to fit a linear model, namely Y = a + b * X1
ii) Using the R build-in function lm(…) to fit a linear model
iii) Print the summary of the model fit

From the summary of model1,

```
Call:
lm(formula = Y ~ X1, data = data)

Residuals:
   Min     1Q Median     3Q    Max
-8.065 -3.107  0.152  3.495  9.587

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -36.9435     3.3651  -10.98 7.62e-12 ***
X1            5.0659     0.2474   20.48  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.252 on 29 degrees of freedom
Multiple R-squared:  0.9353,    Adjusted R-squared:  0.9331
F-statistic: 419.4 on 1 and 29 DF,  p-value: < 2.2e-16
```

The least-squares estimate of the intercept term, a_hat = -36.9435

The least-squares estimate of the coefficient of X_1 , b_hat = 5.0659

Because from this summary, under the column Pr(> |t|), we can see that the p-value associated with the coefficient of X_1 is less than 2e-16, so there is significant evidence that X1 should be kept in the model, i.e. X_1 can explain the variation of Y

R Script Q1b##################

```
# use R build-in function lm(...)
model1 = lm(Y ~ X1, data = data)

# see the summary of model1
summary(model1)
```

##

c) Using R, fit a new model (model 2) in which Y is the response variable, both $X1$ and $X2$ are predictor variables, and there is an intercept term. Is there evidence that model 2 fits the data significantly better than model 1, or not?

please also see the R script below

Outlines:

i) We would like to fit a linear model, namely Y = a + b1 * X1 + b2 * X2

ii) Using the R build-in function lm(…) to fit a linear model
iii) Print the summary of the model fit

From the summary of model2,

```
Call:
lm(formula = Y ~ X1 + X2, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-6.4065 -2.6493 -0.2876  2.2003  8.4847

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -57.9877     8.6382  -6.713 2.75e-07 ***
X1            4.7082     0.2643  17.816  < 2e-16 ***
X2            0.3393     0.1302   2.607   0.0145 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.882 on 28 degrees of freedom
Multiple R-squared:  0.948,     Adjusted R-squared:  0.9442
F-statistic:   255 on 2 and 28 DF,  p-value: < 2.2e-16
```

Using the summary of the model fit, we can see that the coefficient of X_1 is still highly significant (p-value < 2e-16). However, because the p-value associated with the coefficient of X_2 is not significantly small. So, to keep the term X_2 or not depends on the choice of significance level.

Note that the hypothesis to test on a single parameter:

H0: b2 = 0

H1: b2 $\neq$ 0

And we reject H0 if the p-value is “small”, and claim that the explanatory variable X_2 is useful in predicting the response variable Y when all the other variables are included in the model.

For some commonly used significant levels: like 0.05, 0.01

If we choose significance level = 0.05, then we reject H0, and will claim that X_2 is useful in predicting the response variable Y when the constant term and X1 are included in the model;

If we choose significance level = 0.01, then we fail to reject H0, and will claim that X_2 is not useful in predicting the response variable Y when the constant term and X1 are included in the model.

To conclude, to determine whether model2 is better than model1, it depends on the choice of significant levels.

If the significance level = 0.05, then model2 is significantly better than model1

If the significance level = 0.01, then model2 is not significantly better than model1

R Script Q1c##################

```
# use R build-in function lm(...)
model2 = lm(Y ~ X1 + X2, data = data)

# see the summary of model2
summary(model2)
```

##

d) Explore how well model 2 fits the data by considering the residuals.

Please also refer to the below R code to know my code for plotting and finding R^2 and adj R^2

I think there are a few ways to interpret the question by using residuals, I think the question might be asking for:

1) Perform residual diagnostics plots, to see using model2 fitting the data do not violate any assumptions of linear regression (good model fitting should require assumptions satisfaction)

2) Find the coefficient of determination R^2, or find adjusted R^2. (because these quantities are used as a measure of how well the regression model fits the data, and both quantities involved in term called “residuals sum of square”)

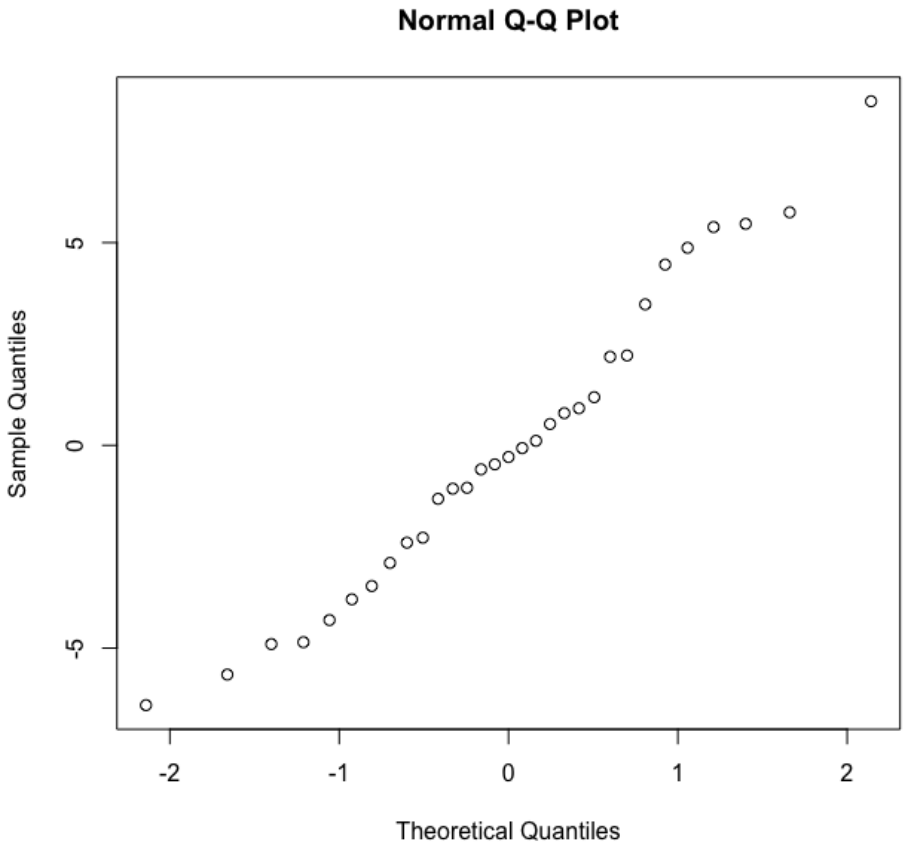
Because I think both methods are also satisfied the question requirements, so I will go for both.

I can just use R to compute because lm(...) method also returns residuals

1) Perform residual diagnostics plots

i) check if the residuals like normal or not

From the plot below, the points almost fall in the straight line, so somehow confirmed normality of residuals



ii) residuals vs predictors

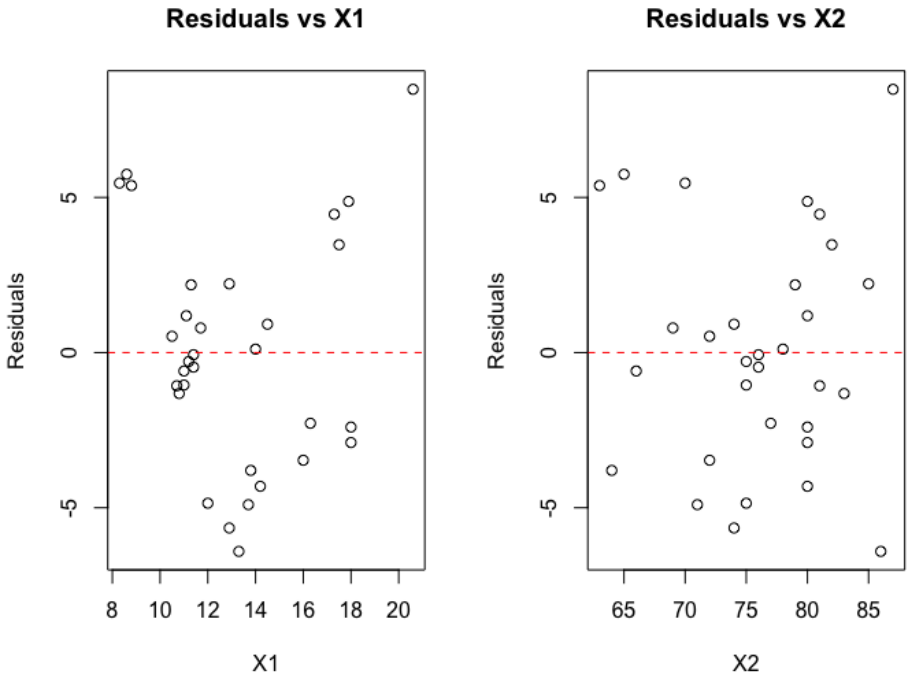
It is expected that residuals should be mean 0 and constant variance

For residuals vs X1 plot:

We can see plot somehow looks like non-linear, because for small and large fitted values, the residuals are positive; for the fitted value between 25 to 45, residuals are negative. But that is not “obvious”

For residuals vs X2 plot:

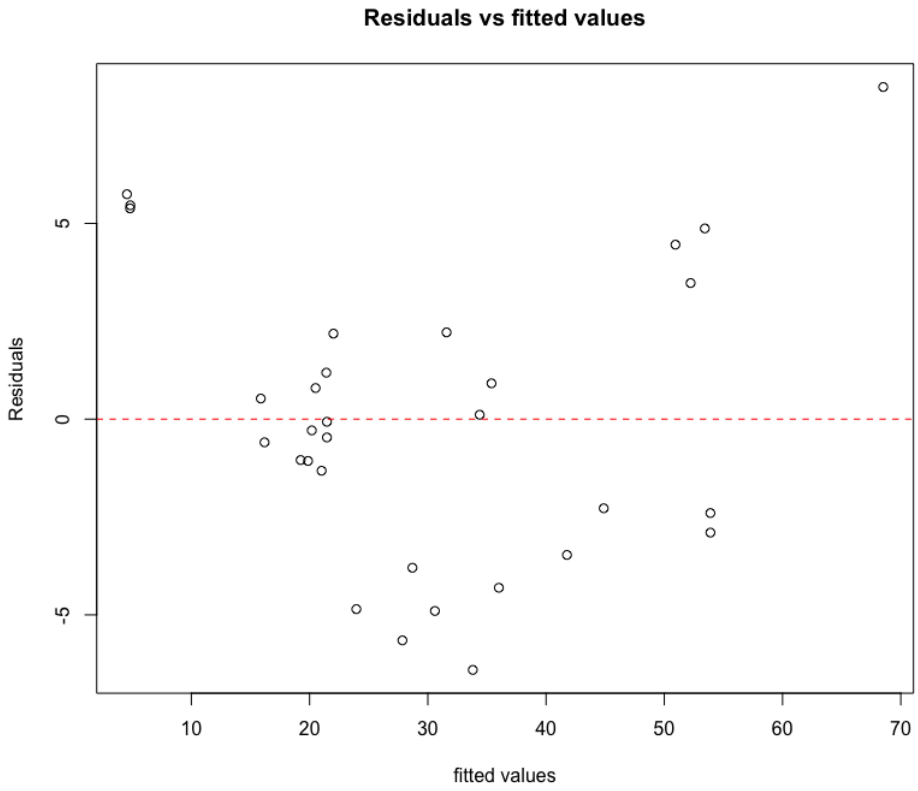
I think this plot looks good.



iii) residuals vs fitted values

It is expected that residuals should be mean 0 and constant variance.

Same situation as in Residuals vs X1. But again, the non-linearity is not obvious



R Script Q1d##################

Q1 d)

residu = model2$residuals

i) find qq plot for residuals to see the residuals like normal

qqnorm(residu)

ii) residuals against fitted values

```
plot(fitted(model2), residu, xlab="fitted values", ylab="Residuals",
    main="Residuals vs fitted values")
abline(h=0, col="red", lty=2)


## iii) residuals against predictors
par(mfrow=c(1, 2))
plot(data$X1, residu, xlab="X1", ylab="Residuals",
    main="Residuals vs X1")
abline(h=0, col="red", lty=2)

plot(data$X2, residu, xlab="X2", ylab="Residuals",
    main="Residuals vs X2")
abline(h=0, col="red", lty=2)


par(mfrow=c(1, 1))



## 2) get residuals sum of square, R^2 and adjusted R^2

RSS = sum(residu^2)


summary(model2)$r.squared
# 0.94795

summary(model2)$adj.r.squared
```

0.9442322

##

2) Find coefficient of determination R^2, or find adjusted R^2

By the definition,

R^2 = 1 - (residuals sum of square) / (total sum of square)

It measures the proportion of variability explained by the regression. And R^2 is often used as a measure of how well the regression models fit the data.

The larger the R^2, the better the fit.

Note that we can also use adjusted R^2 as it accounts for number of parameters

We can just use the R to extract R^2 and adj R^2 from the fitted model2

summary(model2)$r.squared

0.94795

summary(model2)$adj.r.squared

0.9442322

Both values are greater than 90%, so it indicates that the regression model fits the data quite well

To conclude:

From this part, I think “generally” the model2 fit the data quite well. Values of R^2 and adj R^2 are high. Also, the residuals plots generally show some degree of fitness, except the plot for Residuals vs X1 and Residuals vs fitted value, shows mild non-linearity.

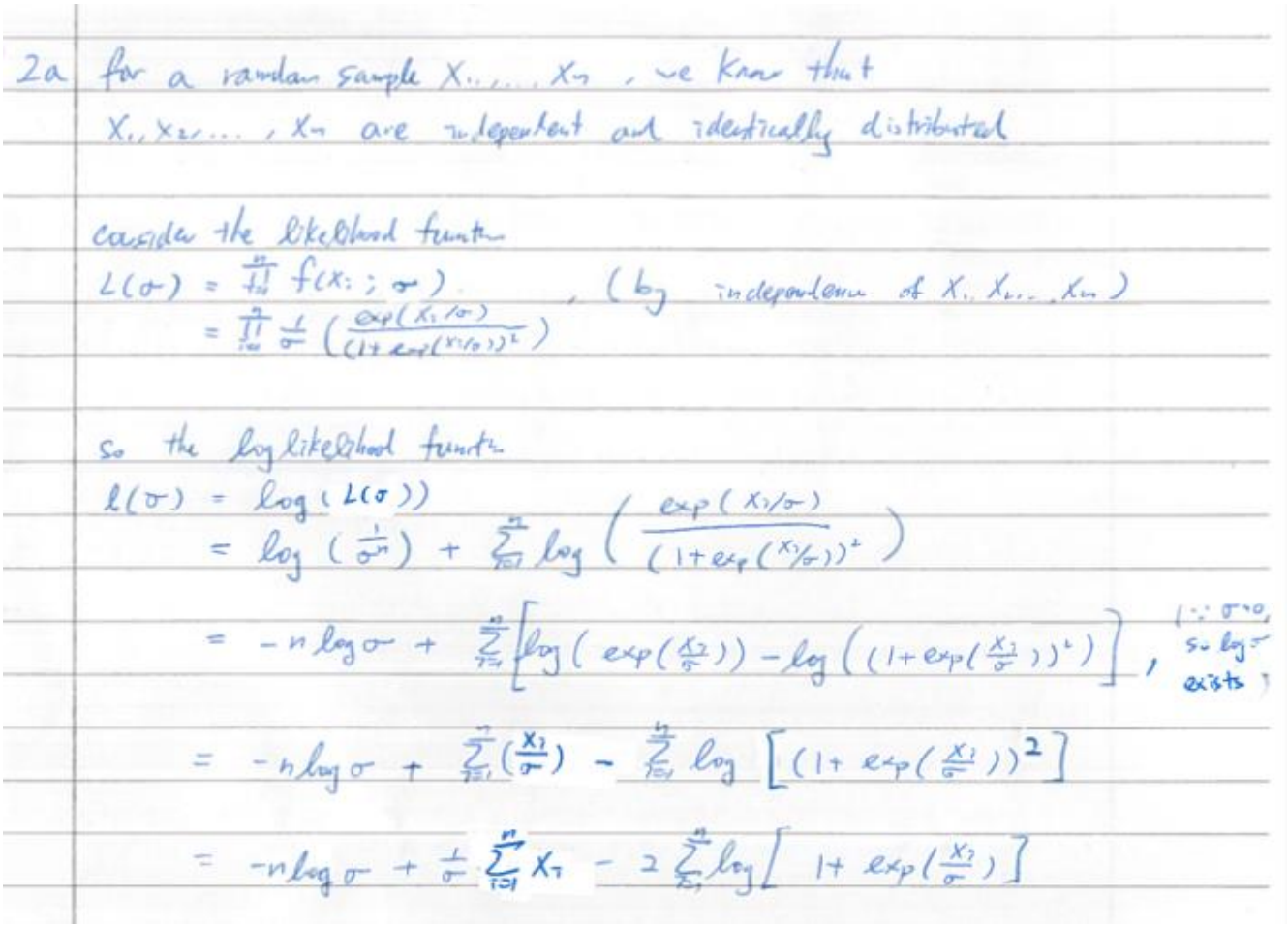
In this case, we may need to perform some transformation.

If we look closer to the raw data, we find that X2 is always at least 4 times X1.

So, transform both X1 and X2 to the range 0 and 1 before fitted into a regression model may be a good trial

Q2

a)



b) please also see the R script for function creation, simple plot, and optimization

Outlines of the method:

i) Declare the function found in 2a)
ii) Plot a simple plot for loglikelihood function on a wide range of σ (say 1e-9 to 500) to ensure that roughly the location of the global maximum. And with initial guess near to the global maximum, and therefore we will not be trapped in local maxima (if any)
iii) Use R build-in function optimize(…) to optimize our loglikelihood function

MLE of σ = 1.44 (to 2 decimal places)

R Script Q2b##################

```
##### Q2 b)
## given data of X
data = c(
  0.337, 0.507, 0.250, 3.131, 6.908,
  0.195, -3.097, 1.002, 2.011, 1.710
)

## next define the loglikelihood function
loglikelihood_function = function(sigma){

  n = length(data)
  sum_of_x = sum(data)

  # first term
  term1 = -1 * n * log(sigma)

  # second term
  term2 = (1/sigma) * sum_of_x

  # third term
  term3 = -2 * sum(log(1 + exp(data / sigma)))

  return (term1 + term2 + term3)
}
```

```
## to ensure global max will be reached, first plot the graph to see if there are any local max
## 0 is excluded because we know that sigma > 0
sigmas = seq(1e-9, 500, by=.1)
logliks = sapply(sigmas, FUN=function(sigma){
  loglikelihood_function(sigma)
})

par(mfrow=c(2, 1))
plot(sigmas, logliks, type="l", main="Plot for looking at global max")
plot(sigmas, logliks, type="l", main="Plot for looking at global max for sigma between 0 and 3", xlim=c(0, 3))
par(mfrow=c(1, 1))
## from the plot, we can see that there are only one maximum,
## and it is located between 0 and 3
## so we are safe to use optimize in the interval (1e-9, 3)

## because we are optimizing one single parameter
## so we need to use optimize

optimize_result = optimize(loglikelihood_function, lower = 1e-9, upper = 3, maximum = TRUE)

mle_sigma = optimize_result$maximum
mle_sigma    # 1.437023
```

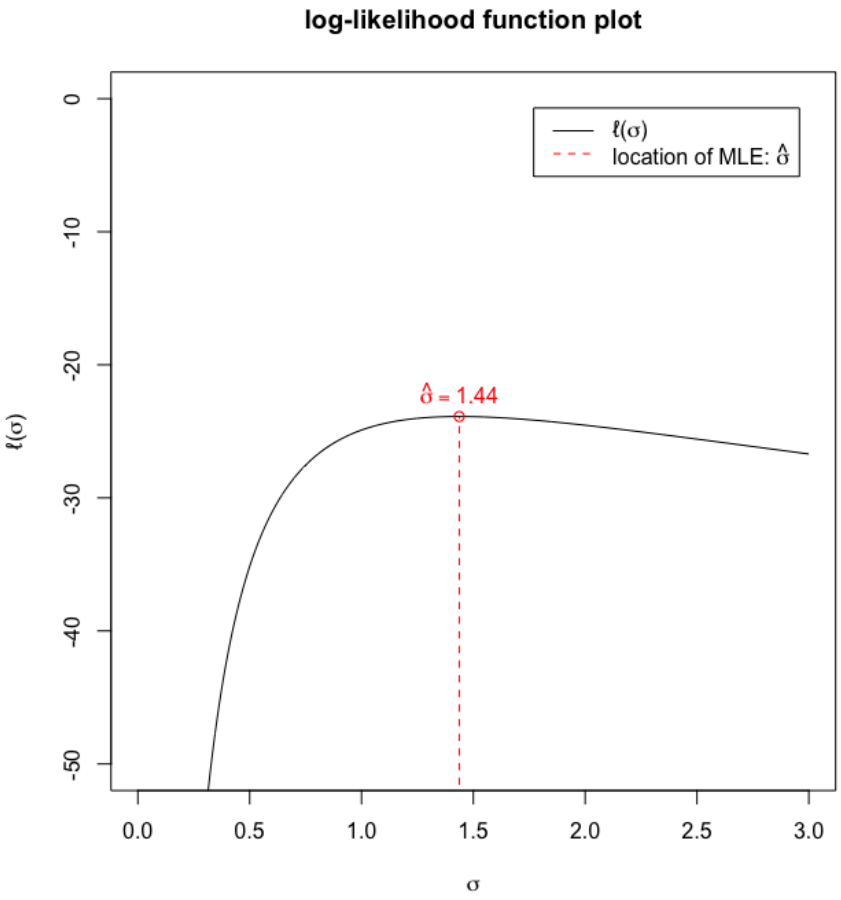
##

c) please also see the R script below for plotting purposes

Produce a plot of $l(\sigma)$ over a suitable range of values with the maximum likelihood estimate of σ

Outlines of the method:

i) Similar to Q2b outline step 2, with narrowing the range of x and y-axis, so the plot will look better
ii) Add necessary labels and legend



R Script Q2c

Q2 c)

```
max_loglik = optimize_result$objective



## plot the loglikelihood function, with sigma between 0 to 5
sigmas = seq(1e-9, 3, length=1000)
logliks = sapply(sigmas, FUN=function(sigma){
  loglikelihood_function(sigma)
})



# because we know that loglik value goes to -Inf when sigma tends to 0
# and there are only one Global Max
# so we can just look at the range of y to (-50, 0), which will also contain the location of MLE
plot(sigmas, logliks, type="l", ylim=c(-50, 0),
    xlab=expression(sigma),
    ylab=expression(paste("\u2113", "(", sigma, ")")),
    main="log-likelihood function plot")

points(mle_sigma, max_loglik, col="red")
text(mle_sigma, max_loglik, col="red", labels=parse(text = paste0('hat(sigma) == ', round(mle_sigma,
2))), pos = 3)

lines(c(mle_sigma, mle_sigma), c(-100, max_loglik), lty = 2, col="red")

legend(x = "topright",
      inset = .05,
      legend = c(expression(paste("\u2113", "(", sigma, ")")),
            expression(paste("location of MLE: ", hat(sigma)))),
      col=c("black", "red"),
```

lty=c(1, 2))

##

##

Q3

a) please also see the R script below

Outlines of the function:

i) Function with parameter called int.T (to avoid ambiguity with T/F boolean value, I am not going to use T as parameter name)
ii) At the beginning of the code block, first check the input is "valid", i.e. positive integer
iii) If the input is valid, then check also int.T equals to 1 or not. That is because $X_1 = 0$. So, if the input is 1, then immediately return the vector c(0); else go to the step iv)
iv) Use the given relationship, as well as the distribution of epsilon (either –1 or 1, with equal probability 0.5), then calculate the $X_2, X_3, \ldots, X_{int.T}$
v) Combine those together with $X_1 = 0$, into a single vector, and then return

R Script Q3a##################

Q3a

because T is a reserved keyword in R

so I use another name

single_realization = function(int.T){

by the question requirement, input can only be positive integer

```
# so check if the input is an integer AND positive, or else exit with message
if(int.T <= 0 | (int.T != round(int.T))){
  stop(paste("input is not an positive integer: ", int.T))
} else {

  # confirmed positive integer, can go on the logic

  # if the input is exactly 1, then just return a vector that containing X1, i.e., 0
  if(int.T == 1){
    return (c(0))
  } else {
    # execute when the input is positive integer greater than 1

    # first store the initial value
    process = as.numeric(0)



    for(i in 2:int.T){
      # note that the length of process is always >= 1
      # so getting the last element is always valid
      x_current = tail(process, 1)

      # epsilon can either 1 or -1, and is equally likely
      epsilon = sample(x = c(-1, 1), size = 1, prob = c(0.5, 0.5))

      # the given relation
      x_next = 2 - x_current + epsilon

      # append the new item x_next at the end of the process vector
```

```
      process = c(process, x_next)
    }

    return (process)

  }
 }
}
```

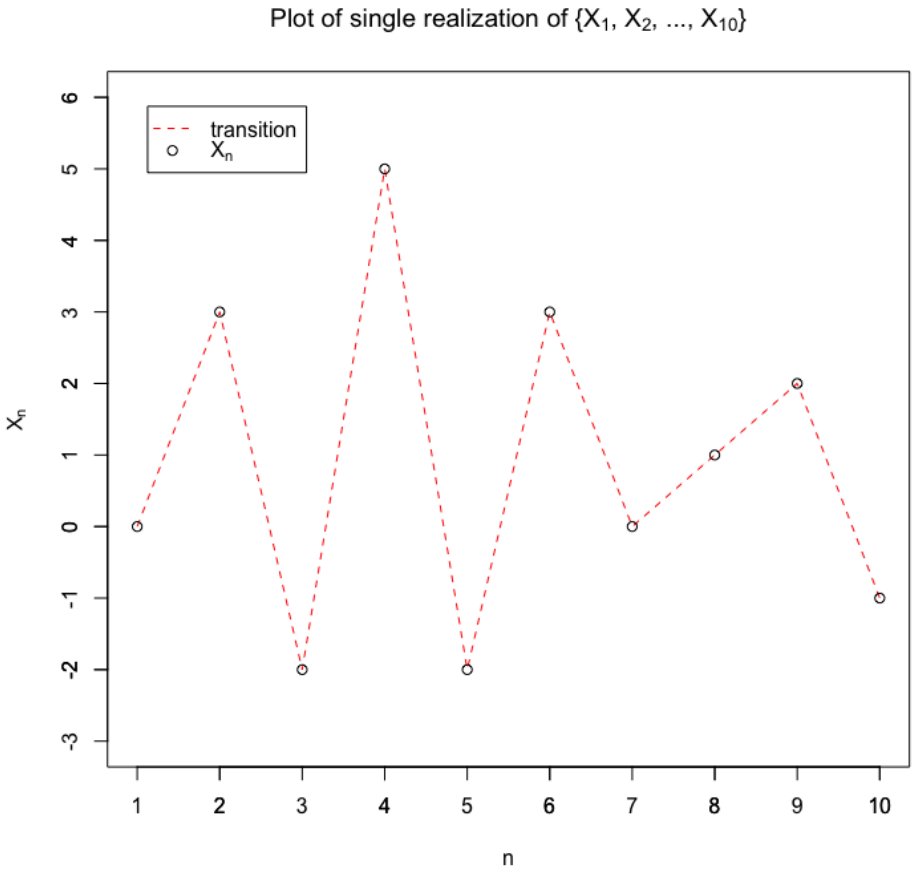
##

b) please also see the R script to see how I make the plot

Produce a plot showing a single realisation of $\{X_1, X_2, \ldots, X_{10}\}$, with the axes suitably labelled.

Steps to get this plot:

i) Using the function created in 3a), with input = 10
ii) x-axis is the step (n = 1, 2, …, 10), y-axis is state of the process at step n (i.e. X_n)
iii) Using R-build it plot function
iv) I used point to indicate X_n , and red-dashed line to indicate the transition of the process in consecutive steps



R Script Q3b##################

```
##### Q3 b)
n = 10
process = single_realization(n)

# plot the line graph as requested
plot(1:n, process, type="l", lty=2, col="red",
    xlim=c(1, n),
```

```
    ylim=c(min(process) - 1, max(process) + 1),    # make the graph a bit wider along y-axis, so an extra +-1 in ylim
    xlab="n",
    ylab=expression(X[n]),
    main=expression(paste("Plot of single realization of {", X[1], ", ", X[2], ", ..., ", X[10], "}")))
points(1:n, process)
axis(1, at = 1:n)
axis(2, at = seq(min(process)-1, max(process)+1, by=1))

legend(x = "topleft", inset = .05,
      legend=c("transition", expression(X[n])),
      lty=c(2, NA),
      pch=c(NA, 1),
      col=c("red", "black"))
```

##

c) Use your function to find numerical estimates of the variance of X_{100} and $P[X_{21} = 0]$. You will not be given marks for using any other method.

For the implementation of the simulation as well as the estimates of the quantities of interest, please also see the R script below

The logic is that we run the function several times (say N = 10000) for the function we created in Q3a), with input = 100

And store all the result into a single matrix, then we will have a 10000 x 100 matrix

Where each row is a simulation result; each col is the transition step

So, to answer the first question Var(X_{100}), find the last column of the simulation result matrix,

Suppose I have a vector named vector_X100, then simply use var(vector_X100)

i) Var(X_{100})

= 99.68786

For the second part, we need to find the proportion of X_{21} = 0 out of N = 10000 simulations, I.e. find

Suppose I have a vector named vector_X21, then find sum(vector_X21==0) / 10000

ii) Pr(X_{21} = 0)

= 0.1787

R Script Q3c##################

```
##### Q3 c)

### first part: find numerical estimates of the variance of X100
### i.e. simulate a couple of times, with input = 100, and then find the last element
### finally calculate the variance

# number of simulation
N = 10000
```

```
# we want to find input n = 100
n = 100



# we will then obtain 10000 rows and 100 columns matrix


# N for number of simulation
# n is the last time step
sim = function(N, n){
  m = as.numeric()


  for(i in 1:N){
    realization = single_realization(n)
    m = rbind(m, realization)
  }


  colnames(m) = 1:n
  rownames(m) = 1:N


  m
}


# after that, perform the simulation as per request
sim_result = sim(N, n)


# find the variance of X100, i.e. the last columns
vector_X100 = sim_result[, 100]
var(vector_X100)
# 99.68786
```

```
# to find P(X21 = 0), we first obtain a vector of X21, then sum if X21 == 0, and finally divided by N


vector_X21 = sim_result[, 21]


# find the ratio

sum(vector_X21 == 0) / N

# 0.1787
```

##