Assignment – 1
STAT 673
Submitted by Radhakrishnan Ravi Vignesh

Exercise 2.1:

i) FitTemp Summary:

lm(formula = data ~ c(1:length(data)), data = data)

Residuals:

Min	1Q	Median	3Q	Max
-0.45577	-0.11892	-0.00059	0.11220	0.37683

Coefficients:

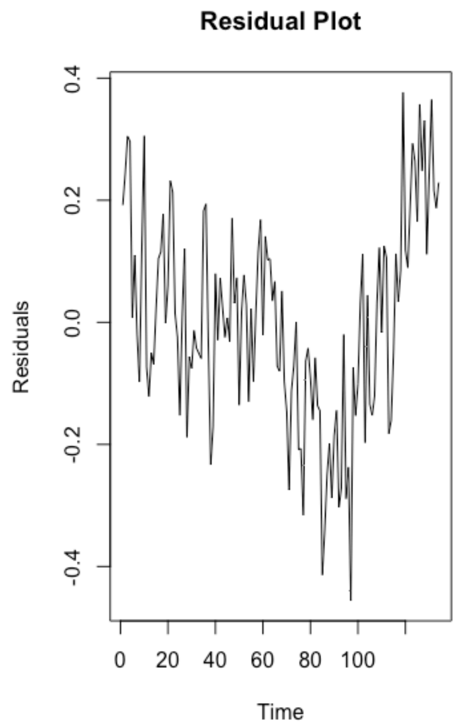
	Estimate	Std. Error	T value	Pr(>|t|)
Intercept	-0.5605117	0.0293850	-19.07	<2e-16 ***
Time	0.0085184	0.0003777	22.55	<2e-16 ***

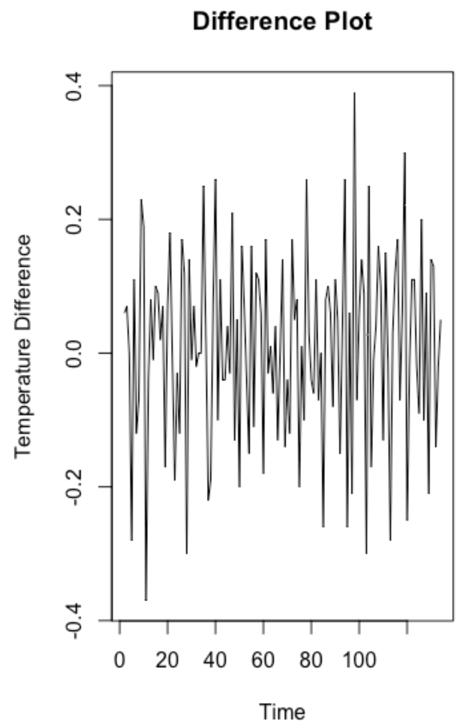
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

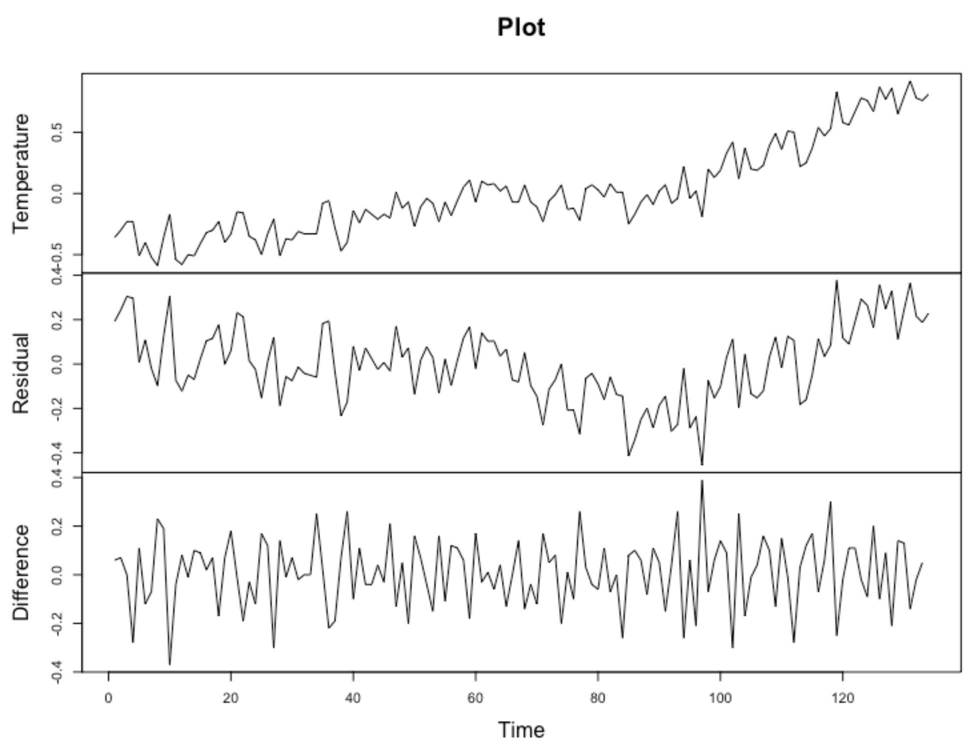
Residual standard error: 0.1691 on 132 degrees of freedom
Multiple R-squared: 0.794, Adjusted R-squared: 0.7924
F-statistic: 508.6 on 1 and 132 DF, p-value: < 2.2e-16

ii) The standard errors in R are reported with the assumption that the residuals are iid and normally distributed. Since the residuals in this time series are correlated, it may not be reliable.

iii) Plot of Residuals and Differences:







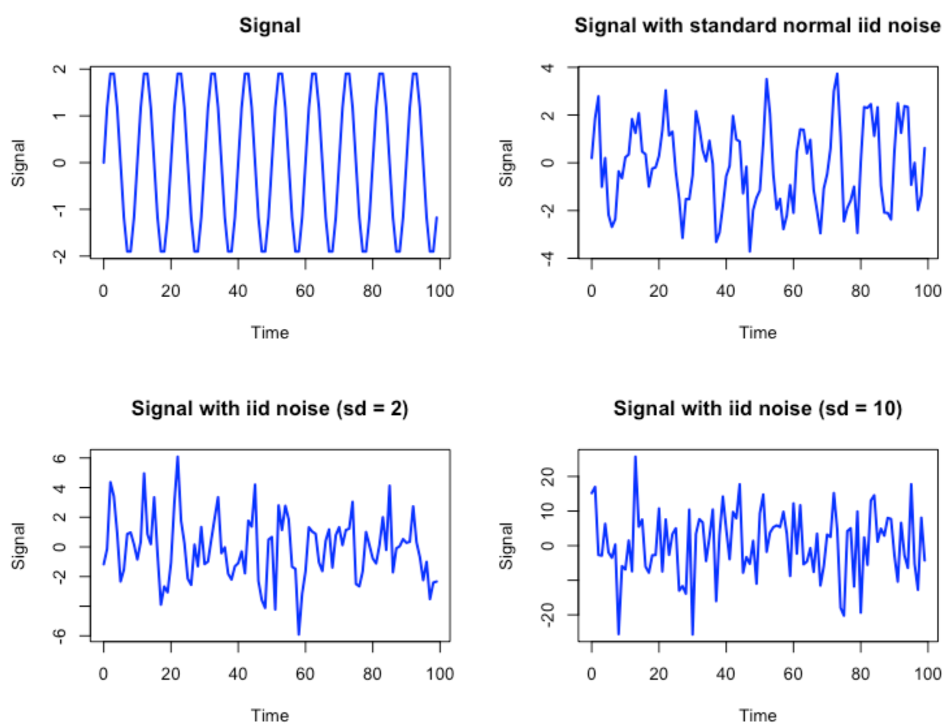
iv) We are able to conclude from the above plots that the residual plot is correlated (but not linearly throughout the plot. It looks to have negative correlation until 100 and a positive correlation after that). The difference plot looks to have uncorrelated data.

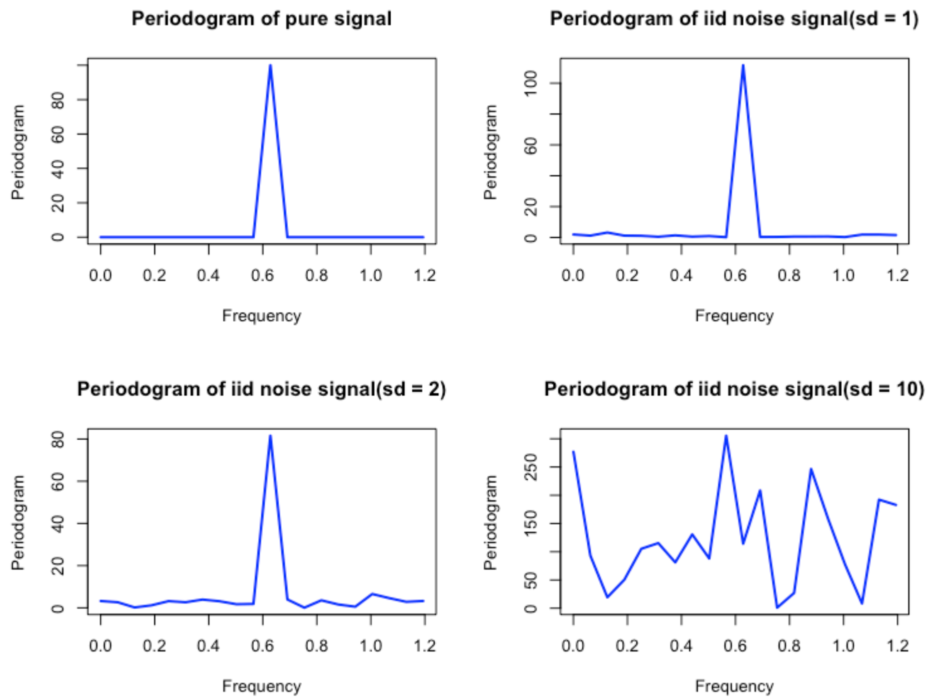
Since the difference plot is effectively a plot of $\beta_0 + \varepsilon_{t+1} - \varepsilon_t$, the difference between errors are uncorrelated

Code:

```
data <-
read.csv('https://www.stat.tamu.edu/~suhasini/teaching673/Data/global_mean_temp.txt',
header = FALSE)
colnames(data) <- c("Temperature")
Time <- c(1:length(data))
data <- ts(data, frequency = 1, start = c(1, 1))
plot.ts(data)
FitTemp <- lm(data ~ Time, data = data)
out = summary(FitTemp)
out
#data_residual <- resid(FitTemp)
data.residual <- data - 0.0085184*c(1:length(data)) + 0.5605117
data_diff <- diff(data, lag = 1)
plot.ts(data_residual, main = "Residual Plot", ylab = "Residuals")
plot.ts(data_diff, main = "Difference Plot", ylab = "Temperature Difference")
df <- data.frame("Data" = data, "Residual" = data_residual, "Difference" = c(data_diff,
NaN))
plot.ts(df, main = "Plot")
```

Exercise 2.3:





A sinusoidal curve with period, 10 is used for this analysis. There is a sharp increase in the periodogram of the signal at its frequency with no noise and when the standard deviation of the noise is increased, the periodogram is disturbed.

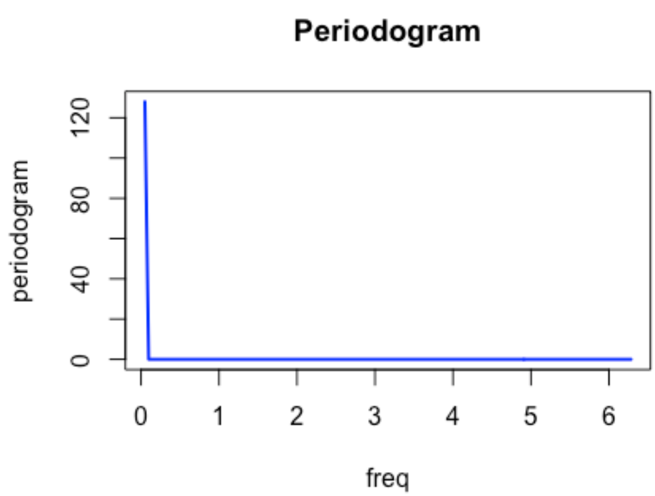
Code:

```
signal = rep(2*sin(2*pi*c(0:9)/10),10)

# Signal plus noise
noisy_signal1 = signal + rnorm(100)
noisy_signal2 = signal + rnorm(100, sd = 2)
noisy_signal3 = signal + rnorm(100, sd = 10)

n = length(signal)
freq = 2*pi*c(0:(n-1))/n

periodogram_signal = (abs(fft(signal))**2)/n1
periodogram_noisy_signal1 = (abs(fft(noisy_signal1))**2)/n1
periodogram_noisy_signal2 = (abs(fft(noisy_signal2))**2)/n1
periodogram_noisy_signal3 = (abs(fft(noisy_signal3))**2)/n1

period1 = n/which.max(periodogram_signal[c(1:20)])
period2 = n/which.max(periodogram_noisy_signal1[c(1:20)])
period3 = n/which.max(periodogram_noisy_signal2[c(1:20)])
period4 = n/which.max(periodogram_noisy_signal3[c(1:20)])
period1
period2
```

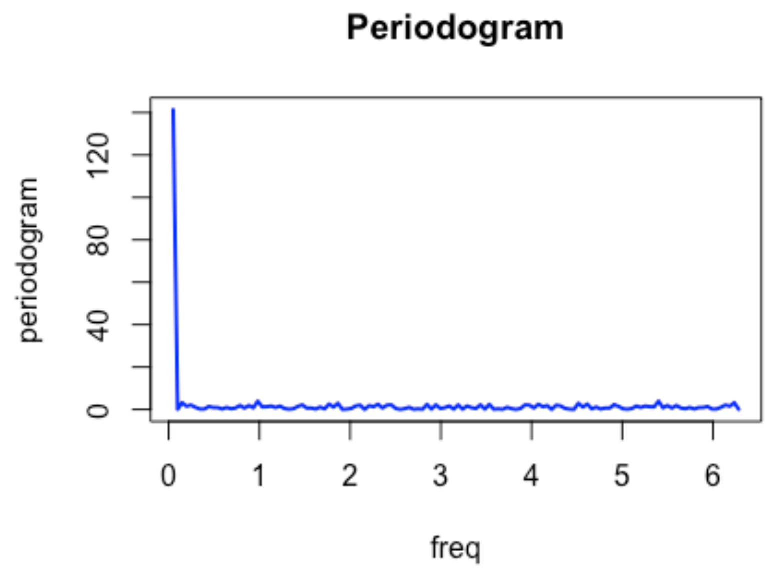
```
period3
period4
time = c(0:(n-1))

par(mfrow=c(2,2))
plot(time,signal,lwd=2,type="l",col="blue",main="Signal", xlab = "Time", ylab = "Signal")
plot(time,noisy_signal1,lwd=2,type="l",col="blue",main="Signal with standard normal iid noise", xlab = "Time", ylab = "Signal")
plot(time,noisy_signal2,lwd=2,type="l",col="blue",main="Signal with iid noise (sd = 2)", xlab = "Time", ylab = "Signal")
plot(time,noisy_signal3,lwd=2,type="l",col="blue",main="Signal with iid noise (sd = 10)", xlab = "Time", ylab = "Signal")

plot(freq[c(1:20)], periodogram_signal[c(1:20)], lwd=2,type="l",col="blue",main="Periodogram of pure signal", xlab = "Frequency", ylab = "Periodogram")
plot(freq[c(1:20)], periodogram_noisy_signal1[c(1:20)], lwd=2,type="l",col="blue",main="Periodogram of iid noise signal(sd = 1)", xlab = "Frequency", ylab = "Periodogram")
plot(freq[c(1:20)], periodogram_noisy_signal2[c(1:20)], lwd=2,type="l",col="blue",main="Periodogram of iid noise signal(sd = 2)", xlab = "Frequency", ylab = "Periodogram")
plot(freq[c(1:20)], periodogram_noisy_signal3[c(1:20)], lwd=2,type="l",col="blue",main="Periodogram of iid noise signal(sd = 10)", xlab = "Frequency", ylab = "Periodogram")
```

Exercise 2.4:

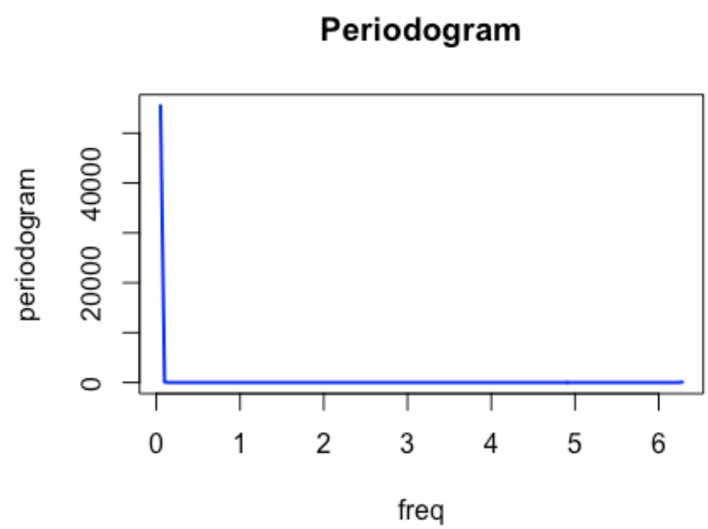
Periodogram of $Y_t = 1$



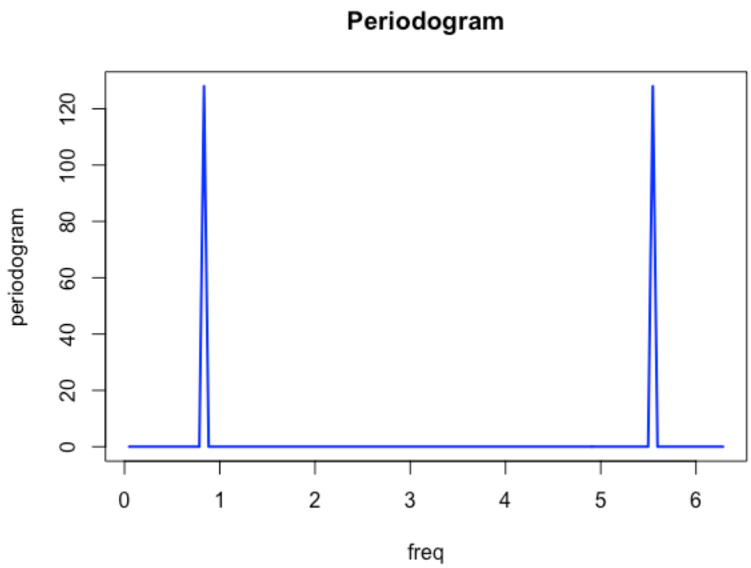
Periodogram of $Y_t = 1+\varepsilon_t$ where the errors are standard normally distributed iid variables



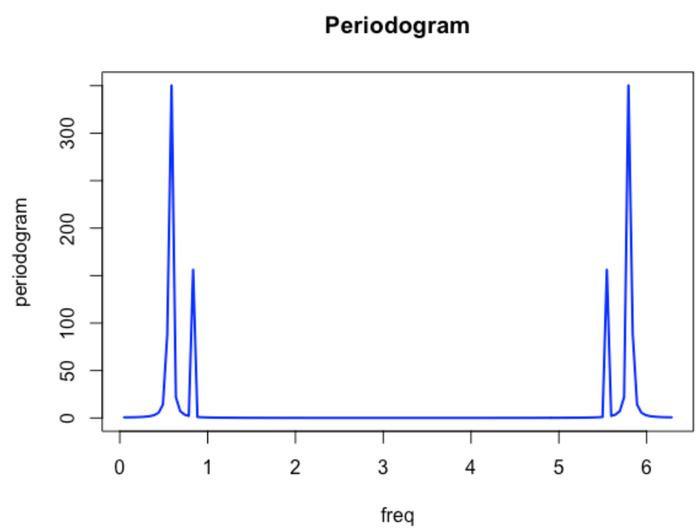
Periodogram of $Y_t = \mu\left(\frac{t}{128}\right) where\ \mu(u) = 5 \times (2u - 2.5u^2) + 20$



Periodogram of $Y_t = 2 \times \sin\left(\frac{2\pi t}{8}\right)$



Periodogram of $Y_t = 2 \times \sin\left(\frac{2\pi t}{8}\right) + 4 \times \cos\left(\frac{2\pi t}{12}\right)$



The maximum occurs at frequency = 0.049 for the first 3 periodograms
The maximum occurs at frequency = 0.834 for the 4^{th} periodogram
The first maximum occurs at frequency = 0.589 for the 5^{th} periodogram since the amplitude of the cosine component is higher.

Code:

```
#Functions
mu <- function(u){
  return(5*(2*u - 2.5*u*u)+20)
```

```
}

plot_periodogram <- function(x) {
  freq = (2*pi*c(0:length(x)-1)/length(x))[1:10]
  periodogram = (abs(fft(x))**2/length(x))[1:10]
  plot(freq,periodogram,lwd=2,type="l",col="blue",main="Periodogram")
  return (which.max((abs(fft(x))**2)/length(x))*2*pi/length(x))
}

par(mfrow = c(1,1))
y = rep(1, 128)
plot_periodogram(y)
y1 = y + rnorm(length(y))
plot_periodogram(y1)
y2 = mu(c(1:128)/128)
plot_periodogram(y2)
y3 = 2*sin(2*pi*c(1:128)/8)
plot_periodogram(y3)
y4 = 2*sin(2*pi*c(1:128)/8) + 4*cos(2*pi*c(1:128)/12)
plot_periodogram(y4)
which.max(abs(fft(y4))**2/length(y4))*2*pi/128
```

Exercise 2.6:

The values of A, B and frequency were obtained from the equation in page 42

$$y_t = 2\sin\left(\frac{2\pi t}{8}\right) + \varepsilon_t$$

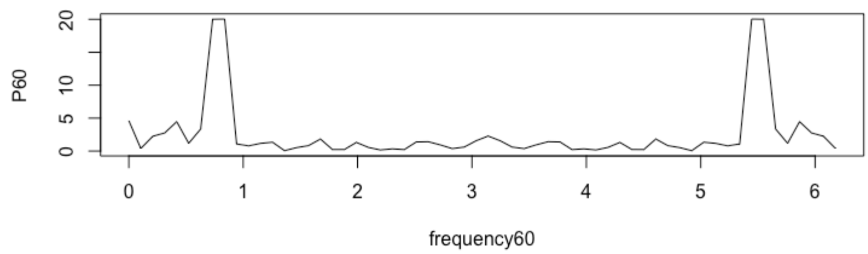
The actual values of A, B, Ω are 0, 2, 2π/8 = 0.78.

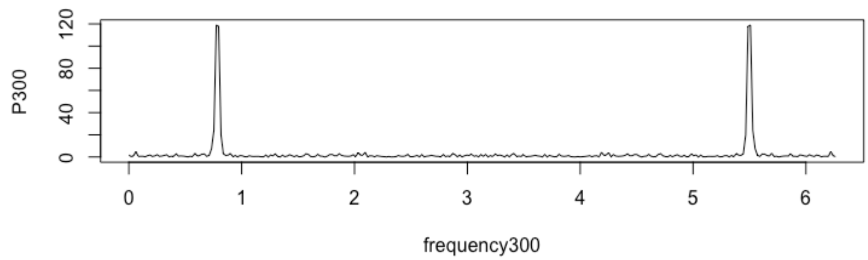
$$\widehat{\Omega}_n = \arg\max_{\omega} I_n(\omega)$$

$$\widehat{A}_n = \frac{2}{n}\sum_{t=1}^{n} Y_t \cos(\widehat{\Omega}_n t) \text{ and } \widehat{B}_n = \frac{2}{n}\sum_{t=1}^{n} Y_t \sin(\widehat{\Omega}_n t).$$

Parts 1 and 2:

The following plot was one of the plots obtained in both the simulations

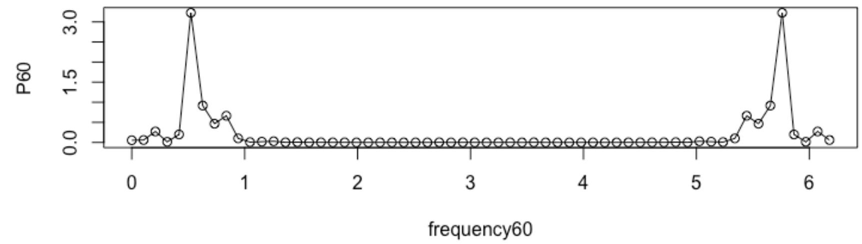


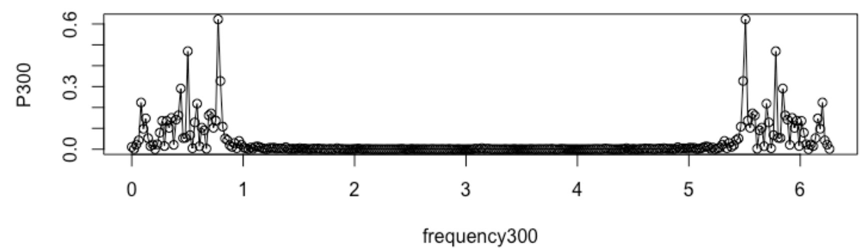


The estimated frequencies were 0.87 and 0.8 in 60-simulation and 300-simulation experiments respectively. So the value of frequency estimated by the 300-simulation experiment was closer to the theoretical frequency which is 0.78. The estimates of A were -0.54 and -0.58 respectively and the estimates of B were -0.61 and -0.6 respectively. The mean squared errors were 2.47 and 2.4 in parts 1 and 2 respectively.

Part 3:

i) The following plot was obtained in one of the simulations in with colored noise





The estimated frequencies were 0.61 and 0.65 in 60-simulation and 300-simulation experiments respectively. The estimate of frequency had more error than in iid noise experiment. The frequency estimated by 300-simulation experiment has given a value closer to 0.78. The mean squared errors were 2.2 and 1.59 for 60-simulation and 300-simulation experiments respectively. The estimates of A were 0.1 and -0.32 respectively and the estimates of B were 0.01 and 0.03 respectively. The mean squared errors are smaller than the error obtained for iid noise.

Code:

```
#Part 1 and 2:

a60 = c()
a300 = c()
b60 = c()
b300 = c()
omegaset60 = c()
omegaset300 = c()


for (i in 1:100){
  signal_60 <- 2*sin(2*pi*c(0:59)/8) + rnorm(60)
  signal_300 <- 2*sin(2*pi*c(0:299)/8) + rnorm(300)

  P60 <- (abs(fft(signal_60))**2)/60
  P300 <- (abs(fft(signal_300))**2)/300
```

```
  frequency60 <- 2*pi*c(0:59)/60
  frequency300 <- 2*pi*c(0:299)/300

  par(mfrow=c(2,1))
  #plot.ts(signal_60)
  #plot.ts(signal_300)

  plot(frequency60, P60,type="l")
  plot(frequency300, P300,type="l")

  omega60 = which.max(P60[c(1:20)])*2*pi/60
  omega300 = which.max(P300[c(1:40)])*2*pi/300


  A60 = 2*sum(signal_60*cos(omega60*c(1:60)))/60
  A300 = 2*sum(signal_300*cos(omega300*c(1:300)))/300

  B60 = 2*sum(signal_60*sin(omega60*c(1:60)))/60
  B300 = 2*sum(signal_300*sin(omega300*c(1:300)))/300

  omegaset60 = c(omegaset60, omega60)
  omegaset300 = c(omegaset300, omega300)

  a60 = c(a60, A60)
  a300 = c(a300, A300)
  b60 = c(b60, B60)
  b300 = c(b300, B300)
}

mean(a60)
mean(a300)
mean(b60)
mean(b300)
mean(omegaset60)
mean(omegaset300)

error60 = sum(c(a60-0, b60-2, omegaset60 - (2*pi/8))*c(a60-0, b60-2, omegaset60 -
(2*pi/8)))/300
error60
error300 = sum(c(a300-0, b300-2, omegaset300 - (2*pi/8))*c(a300-0, b300-2,
omegaset300 - (2*pi/8)))/300
error300

#Part 3:

#colored noise
```

```
a60 = c()
a300 = c()
b60 = c()
b300 = c()
omegaset60 = c()
omegaset300 = c()
for (i in 1:100){
ar2_60 <- arima.sim(list(order=c(2,0,0), ar = c(1.5, -0.75)), n=60)
ar2_300 <- arima.sim(list(order=c(2,0,0), ar = c(1.5, -0.75)), n=300)

signal_60 <- 2*sin(2*pi*c(1:60)/8) + ar2_60
signal_300 <- 2*sin(2*pi*c(1:300)/8) + ar2_300

P60 <- abs(fft(signal_60)/60)**2
P300 <- abs(fft(signal_300)/300)**2

frequency60 <- 2*pi*c(0:59)/60
frequency300 <- 2*pi*c(0:299)/300

par(mfrow=c(2,1))
plot.ts(signal_60)
plot.ts(signal_300)

plot(frequency60, P60,type="o")
plot(frequency300, P300,type="o")

omega60 = which.max(P60[c(1:20)])*2*pi/60
omega300 = which.max(P300[c(1:40)])*2*pi/300

A60 = 2*mean(signal_60*cos(omega60*c(1:60)))
A300 = 2*mean(signal_300*cos(omega300*c(1:300)))

B60 = 2*mean(signal_60*sin(omega60*c(1:60)))
B300 = 2*mean(signal_300*sin(omega300*c(1:300)))

omegaset60 = c(omegaset60, omega60)
omegaset300 = c(omegaset300, omega300)

a60 = c(a60, A60)
a300 = c(a300, A300)
b60 = c(b60, B60)
b300 = c(b300, B300)
}

mean(a60)
mean(a300)
```

```
mean(b60)
mean(b300)
mean(omegaset60)
mean(omegaset300)
error60 = sum(c(a60-0, b60-2, omegaset60 - (2*pi/8))*c(a60-0, b60-2, omegaset60 -
(2*pi/8)))/300
error60
error300 = sum(c(a300-0, b300-2, omegaset300 - (2*pi/8))*c(a300-0, b300-2,
omegaset300 - (2*pi/8)))/300
error300
```