tharunte_Homework3

Question 1

Exhibit 1.1 Time Series Plot of Los Angeles Annual Rainfall

```
library(TSA)
```

```
##
## Attaching package: 'TSA'

## The following objects are masked from 'package:stats':
##
##     acf, arima

## The following object is masked from 'package:utils':
##
##     tar
```

```
# win.graph(width=4.875, height=2.5,pointsize=8)
data(larain)
plot(larain,ylab='Inches',xlab='Year',type='o')
```

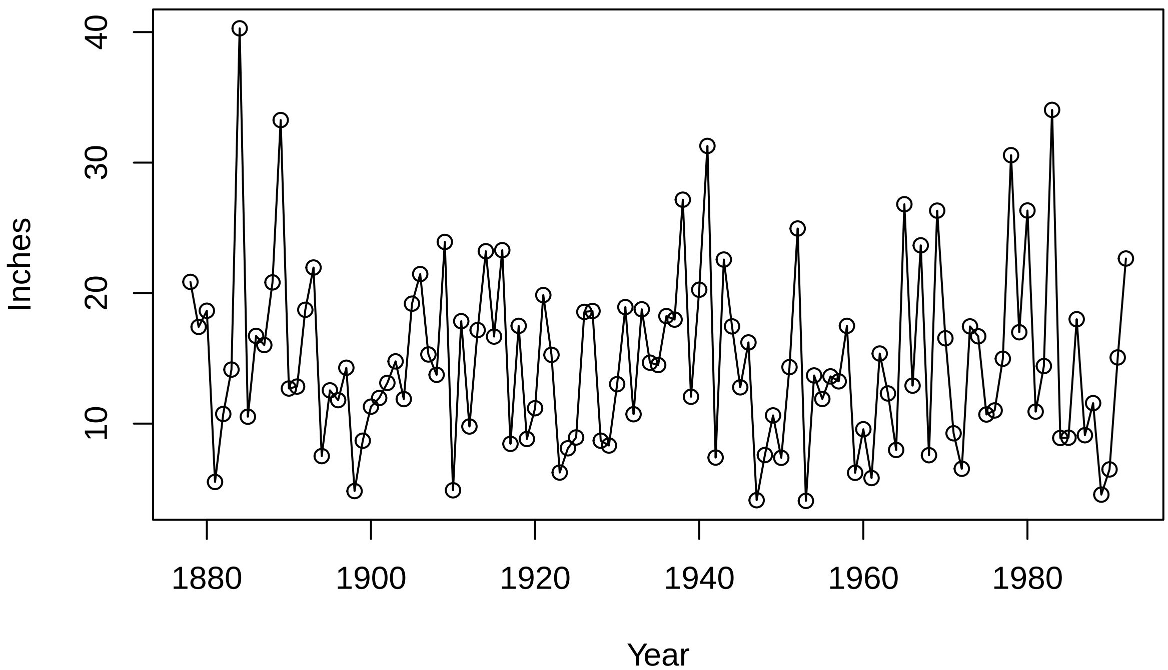


Exhibit 1.2 Scatterplot of LA Rainfall versus Last Year's LA Rainfall

```
# win.graph(width=3,height=3,pointsize=8)
plot(y=larain, x=zlag(larain),ylab='Inches', xlab='Previous Year Inches')
```

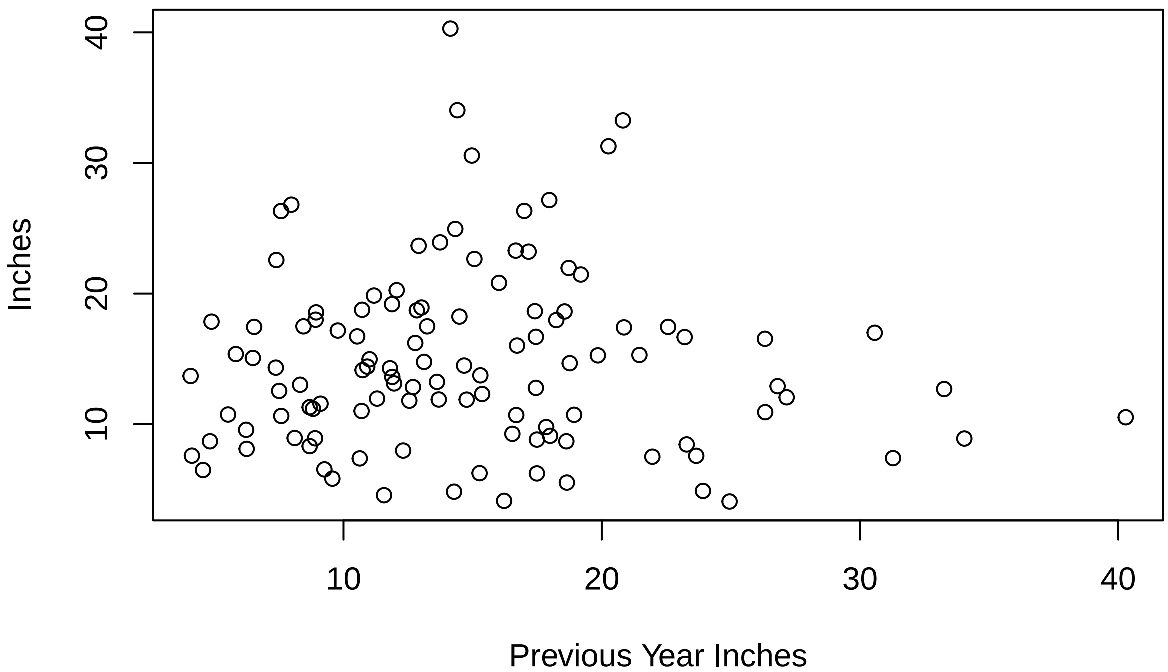


Exhibit 1.3 Time Series Plot of Color Property from a Chemical Process

```
# win.graph(width=5, height=2.5, pointsize=8)
data(color)
plot(color,ylab='Color Property',xlab='Batch',type='o')
```

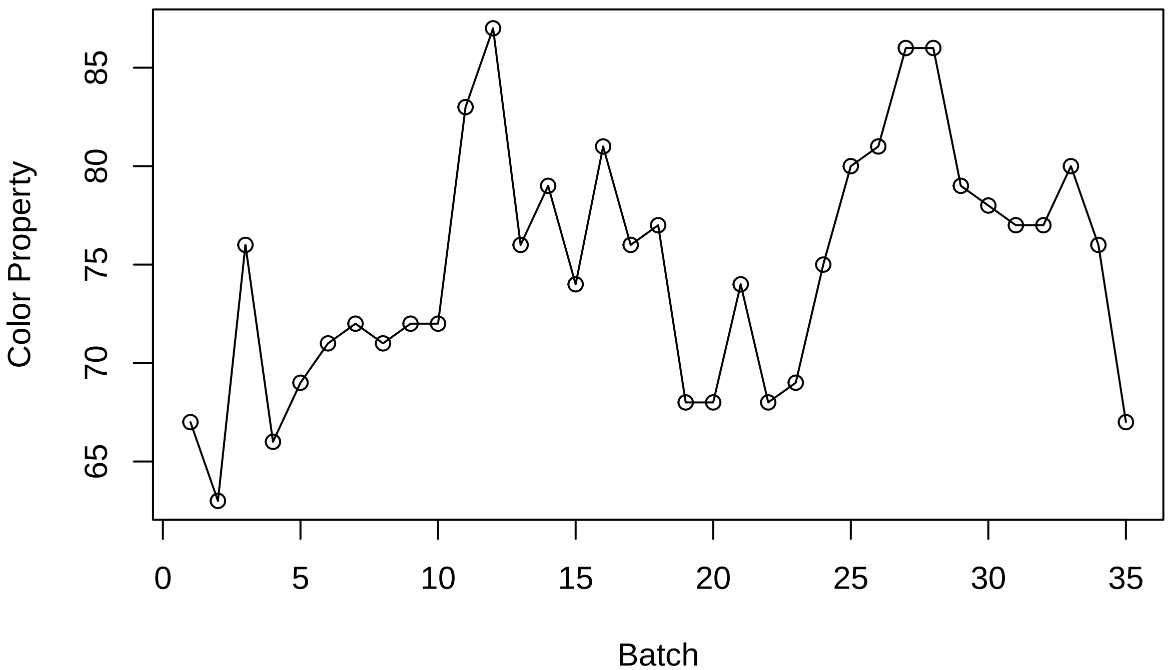
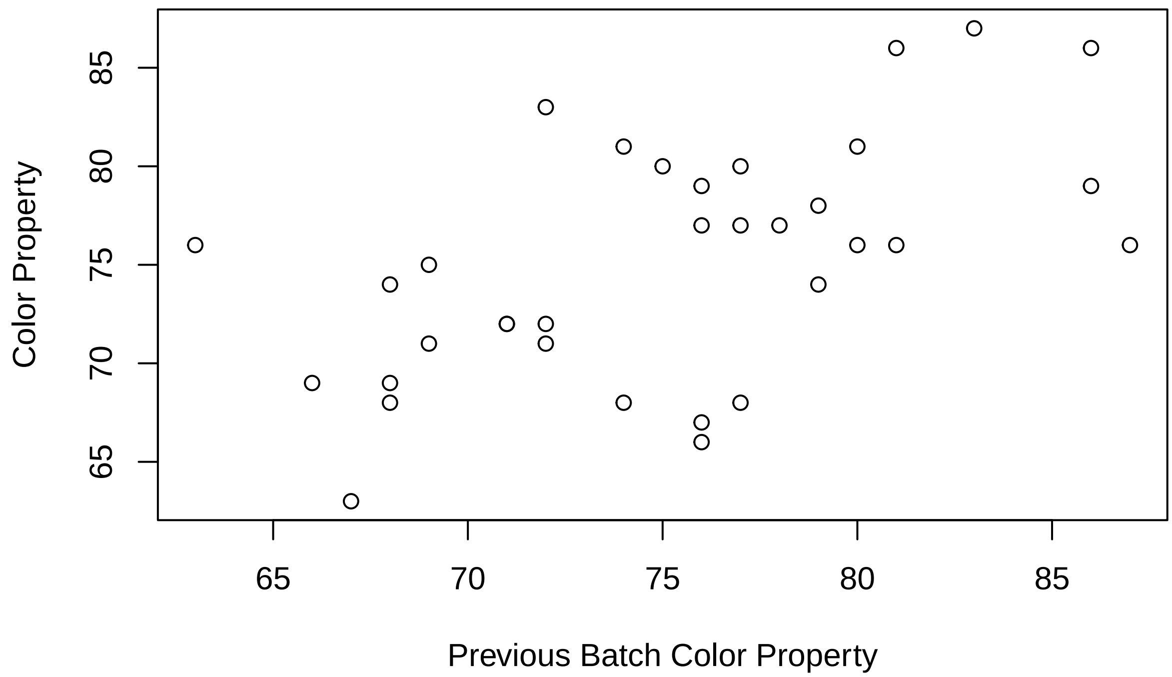


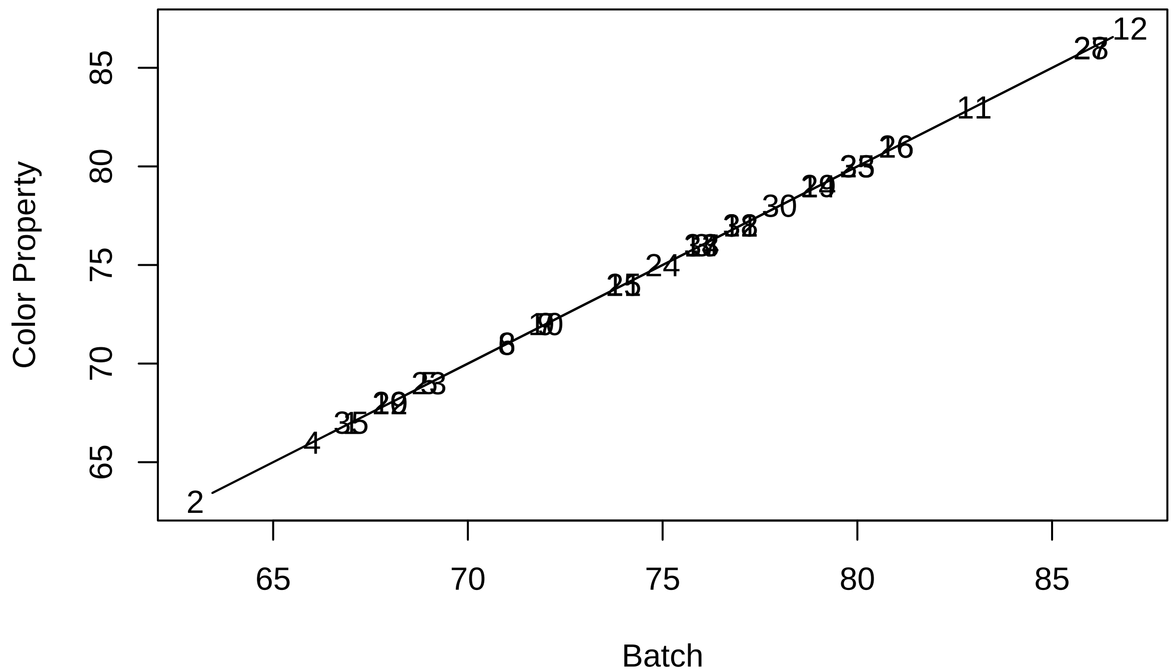
Exhibit 1.4 Scatterplot of Color Value versus Previous Color Value

```r
# win.graph(width=3,height=3,pointsize=8)
plot(y=color, x=zlag(color),ylab='Color Property',xlab='Previous Batch Color Property')
```



pratice running commands suggested in theory

```
plot(color, color, ylab='Color Property',xlab='Batch',type='o')
```




```r
as.vector(color)
```

```
##  [1] 67 63 76 66 69 71 72 71 72 72 83 87 76 79 74 81 76 77 68 68 74 68 69 75 80
## [26] 81 86 86 79 78 77 77 80 76 67
```

```r
plot(as.vector(color), color, ylab='Color Property',
xlab='Batch',type='o')
```

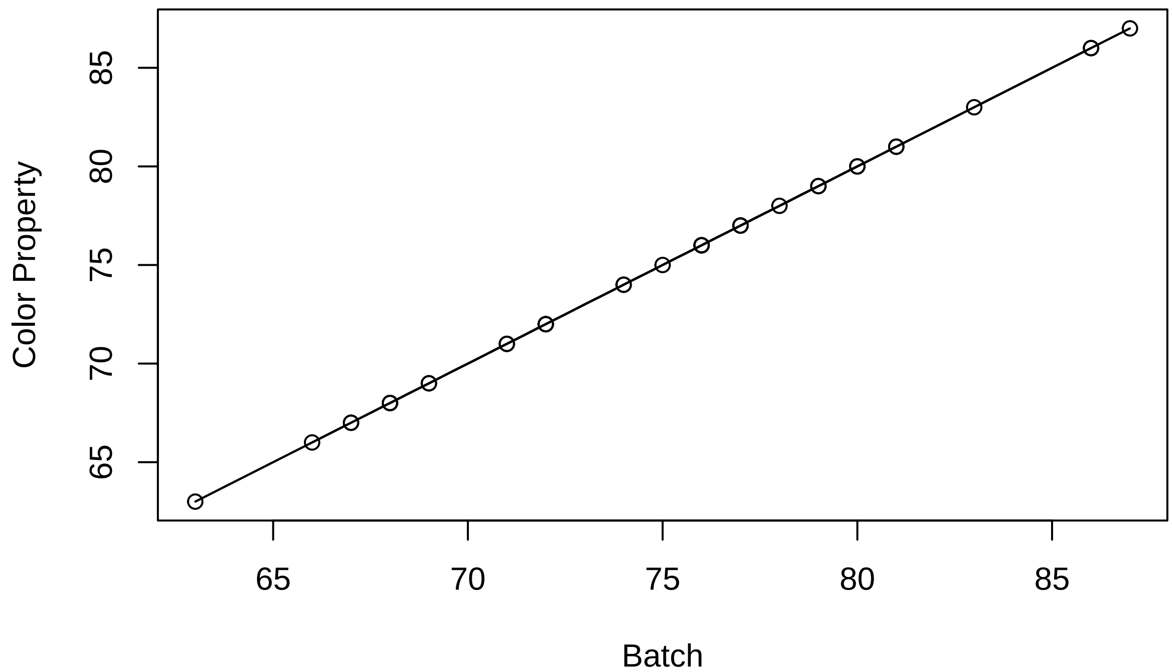


Exhibit 1.5 Abundance of Canadian Hare

```
# win.graph(width=4.875, height=2.5,pointsize=8)
data(hare)
plot(hare,ylab='Abundance',xlab='Year',type='o')
```

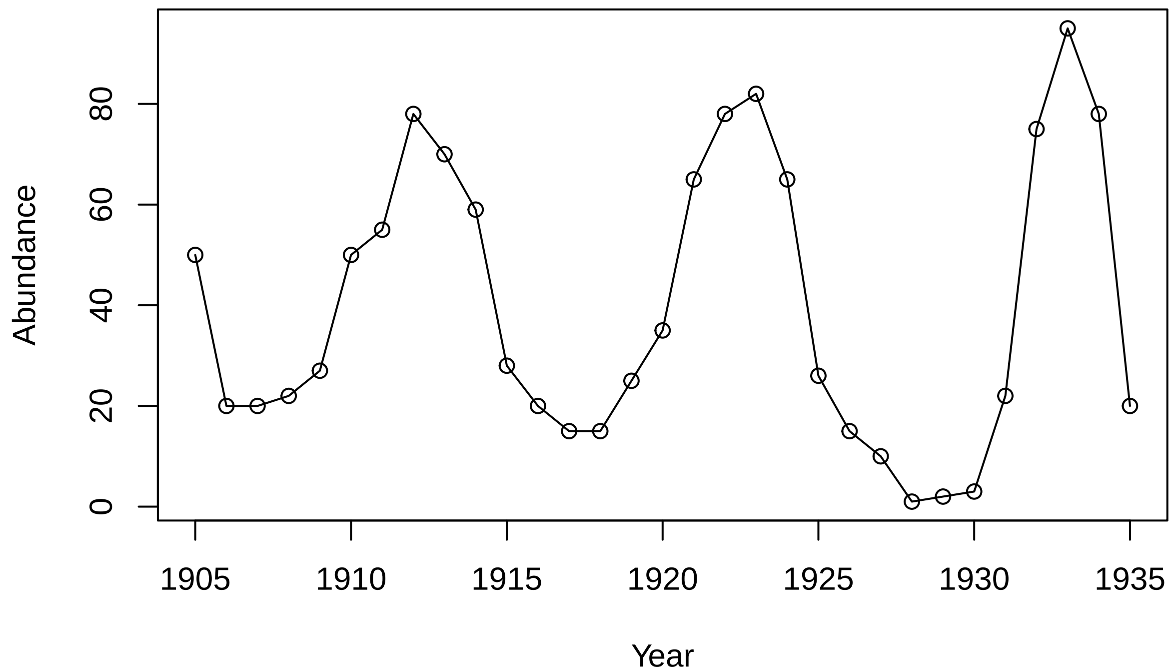
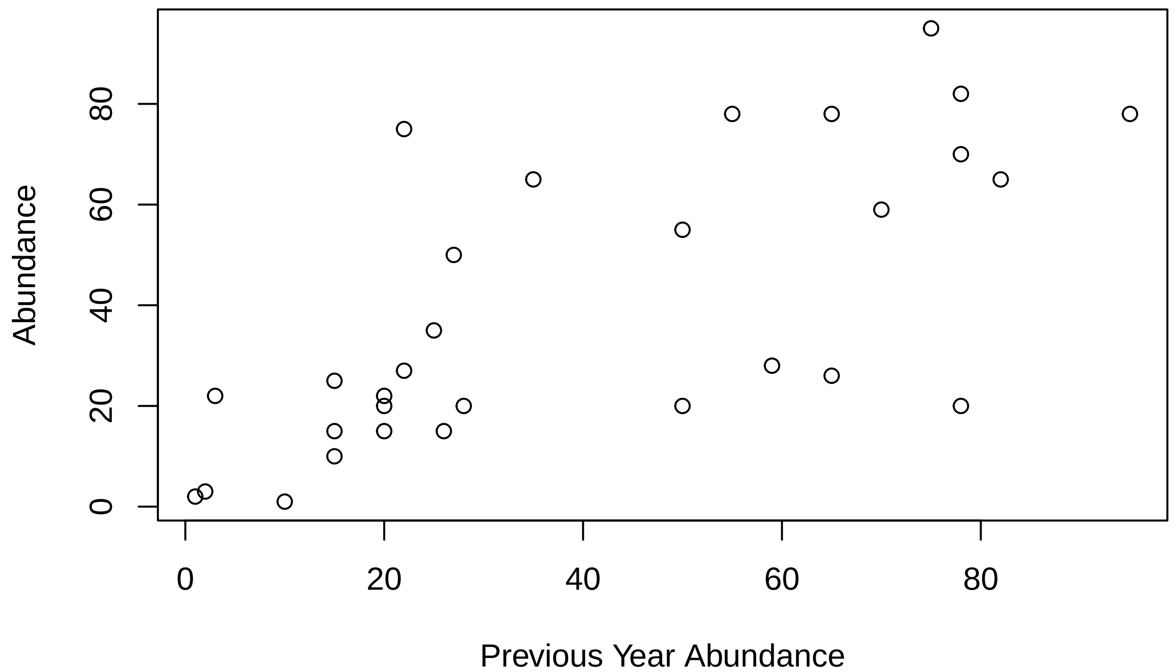


Exhibit 1.6 Hare Abundance versus Previous Year's Hare Abundance

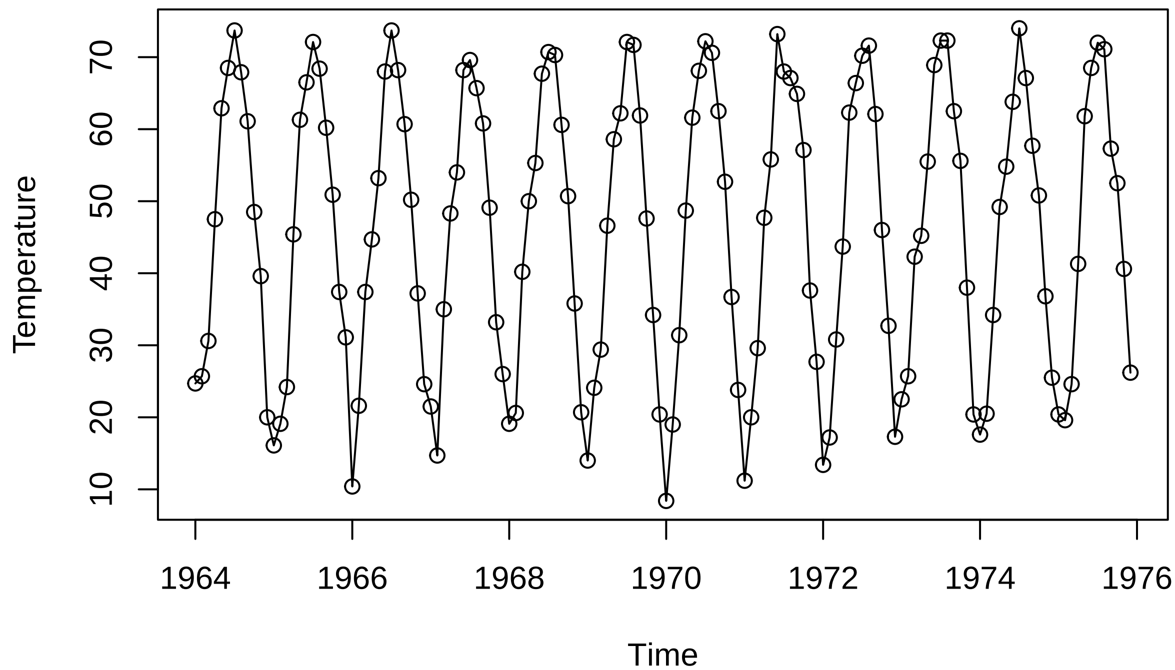
```
# win.graph(width=3, height=3,pointsize=8)
plot(y=hare,x=zlag(hare),ylab='Abundance',xlab='Previous Year Abundance')
```



Monthly Average Temperatures in Dubuque, Iowa

Exhibit 1.7 Average Monthly Temperatures, Dubuque, Iowa

```r
# win.graph(width=4.875, height=2.5,pointsize=8)
data(tempdub)
plot(tempdub,ylab='Temperature',type='o')
```



Monthly Oil Filter Sales

Exhibit 1.8 Monthly Oil Filter Sales

```
# win.graph(width=4.875, height=2.5,pointsize=8)
data(oilfilters)
plot(oilfilters,type='o',ylab='Sales')
```

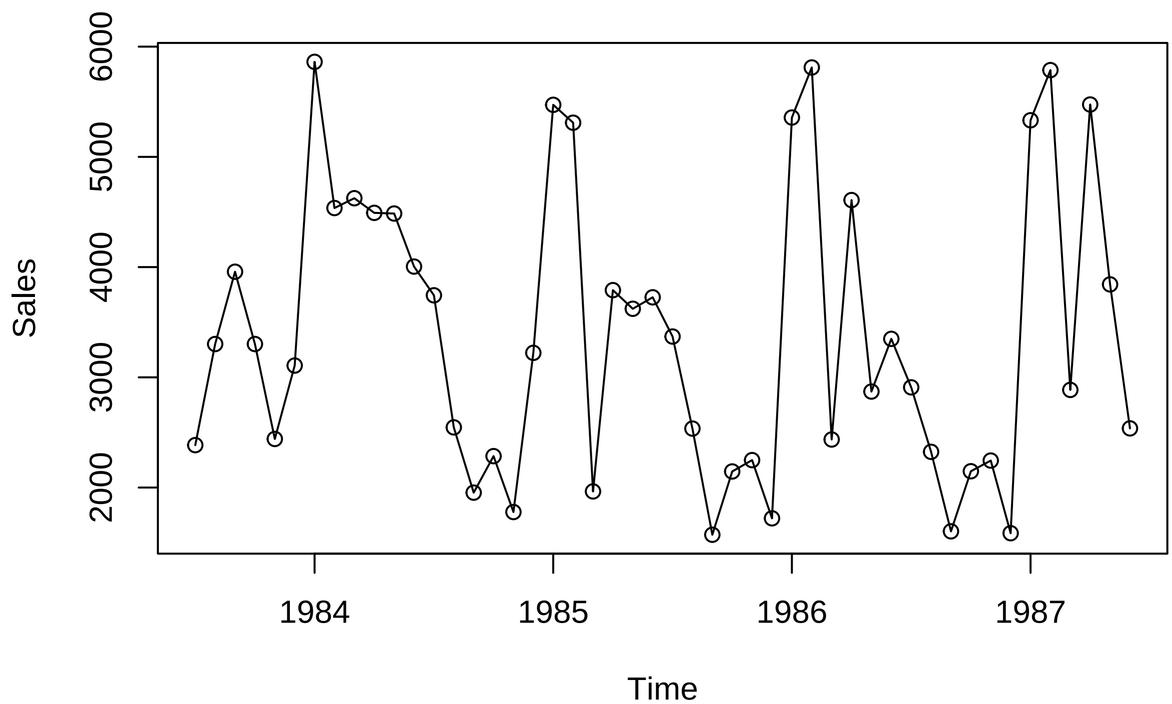
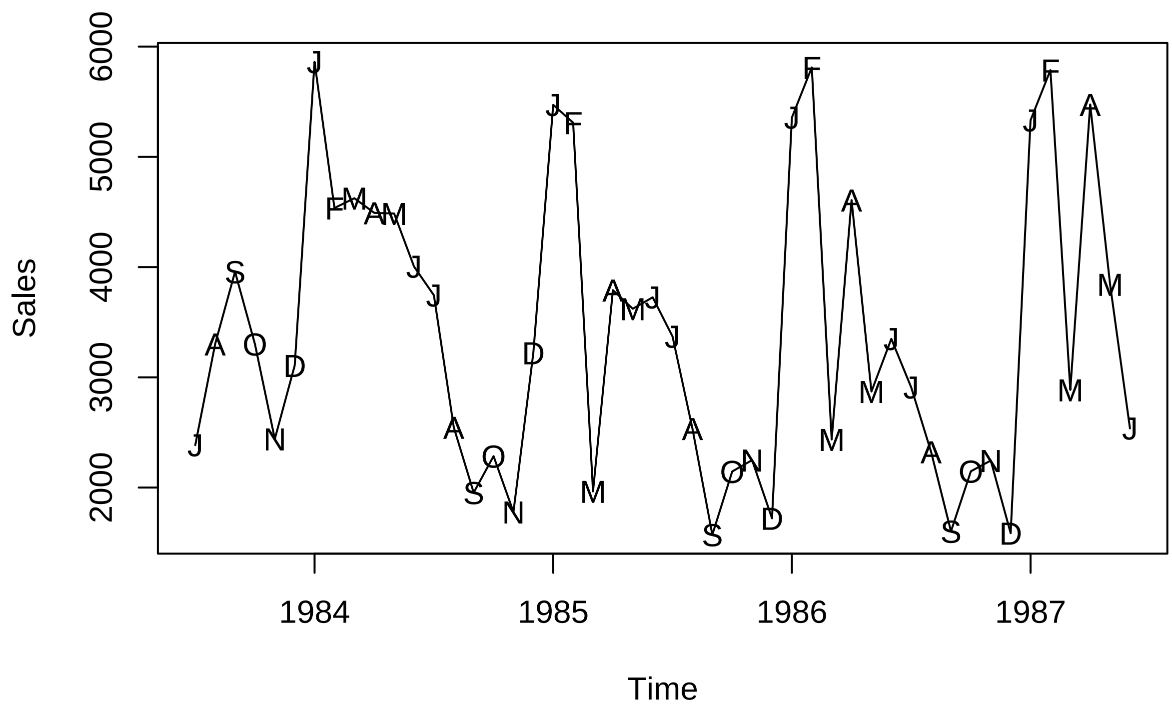
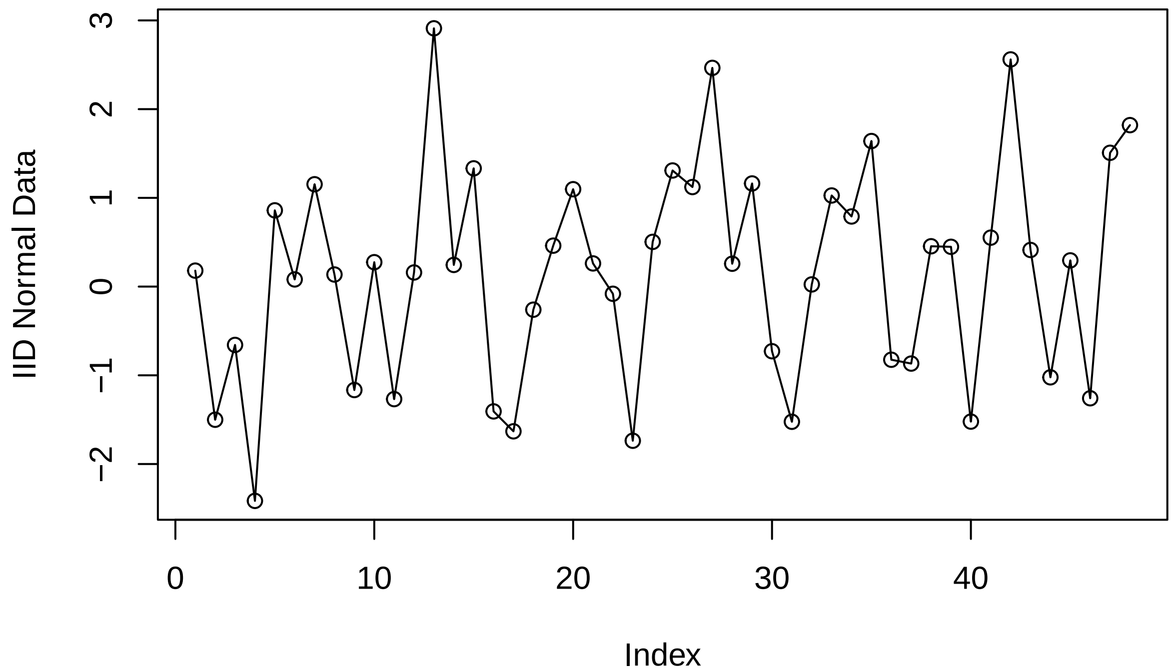


Exhibit 1.9 Monthly Oil Filter Sales with Special Plotting Symbols

```
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(oilfilters,type='l',ylab='Sales')
points(y=oilfilters,x=time(oilfilters),pch=as.vector(season(oilfilters)))
```

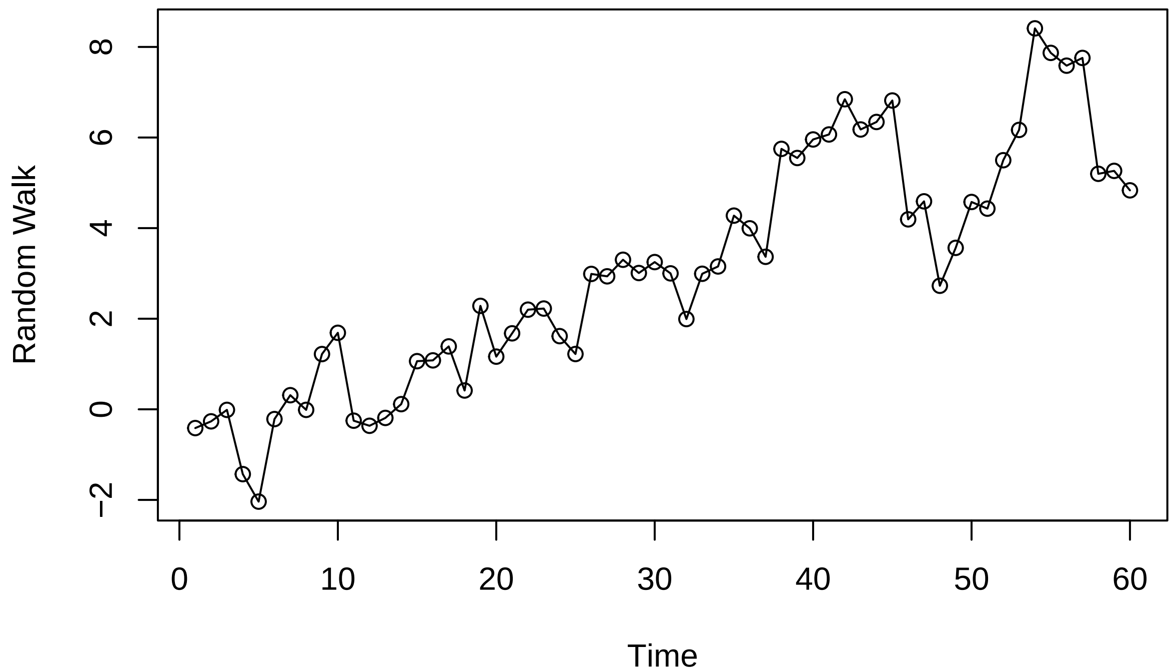



```r
y=rnorm(48)
plot(y, type='o', ylab='IID Normal Data')
```



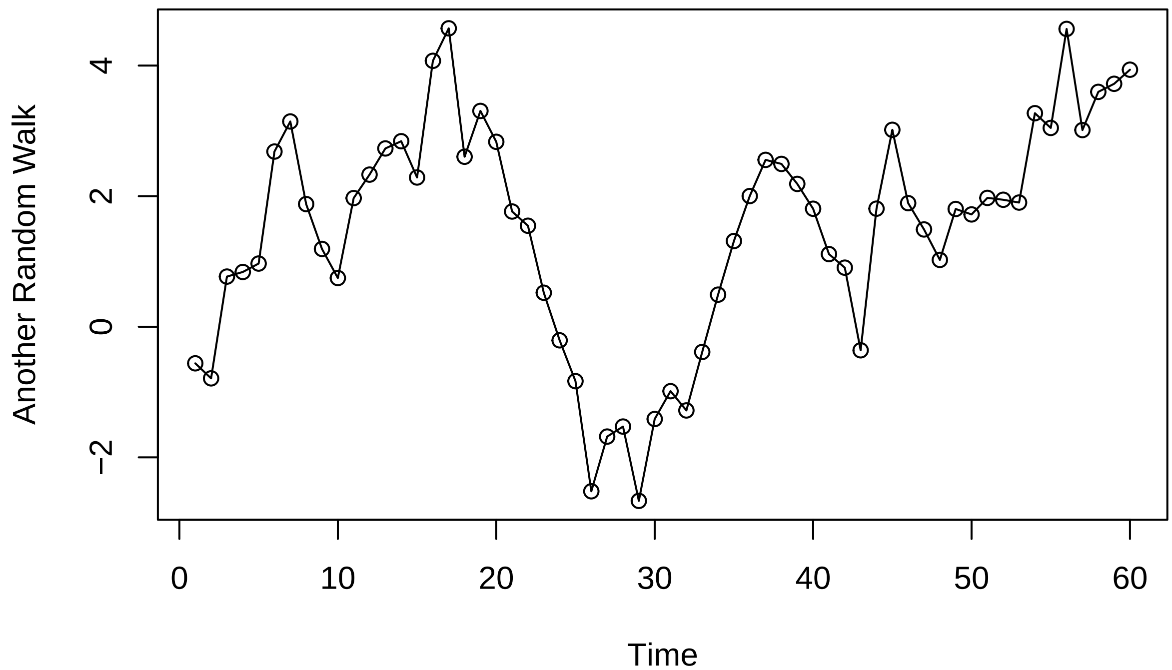
Chapter 2

```r
# win.graph(width=4.875, height=2.5,pointsize=8)
data(rwalk)
plot(rwalk,type='o',ylab='Random Walk')
```



Manually creating a random walk data

```
# win.graph(width=4.875, height=2.5,pointsize=8)
n=60
set.seed(123)
sim.random.walk=ts(cumsum(rnorm(n)),freq=1,start=1)
plot(sim.random.walk,type='o',ylab='Another Random Walk')
```



Chapter 3

Exhibit 3.1 Least Squares Regression Estimates for Linear Time Trend

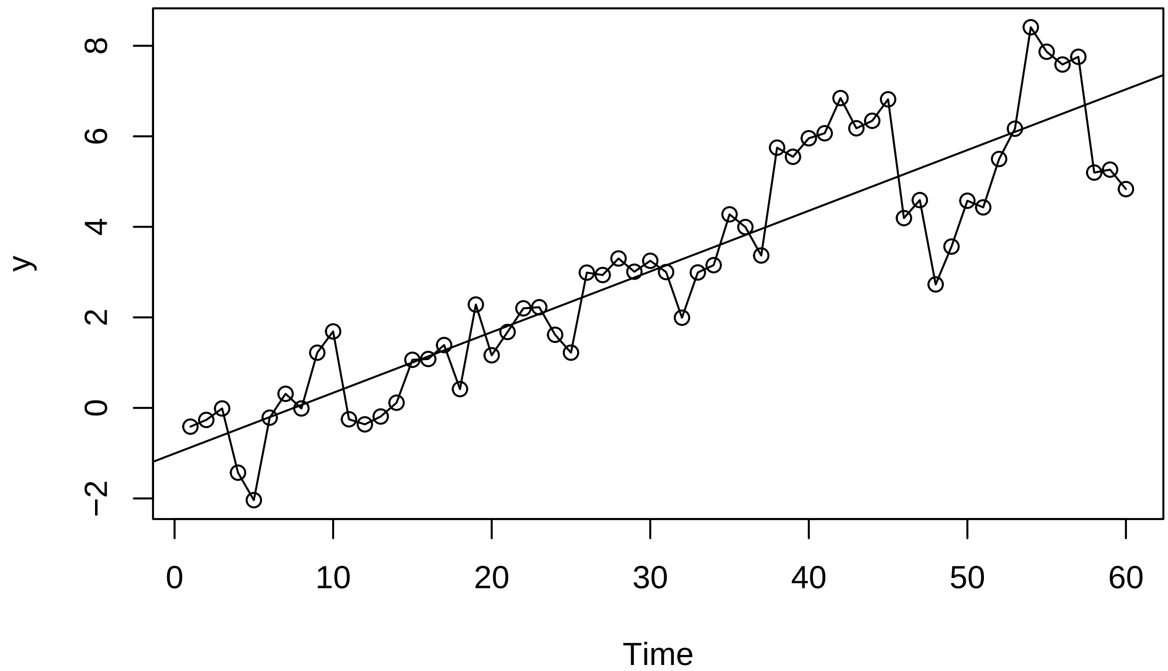
```
data(rwalk)
model1=lm(rwalk~time(rwalk))
summary(model1)
```

```
##
## Call:
## lm(formula = rwalk ~ time(rwalk))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.70045 -0.79782  0.06391  0.63064  2.22128
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.007888   0.297245  -3.391  0.00126 **
## time(rwalk)  0.134087   0.008475  15.822  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.137 on 58 degrees of freedom
## Multiple R-squared:  0.8119, Adjusted R-squared:  0.8086
```

```
## F-statistic: 250.3 on 1 and 58 DF,  p-value: < 2.2e-16
```

Exhibit 3.2 Random Walk with Linear Time Trend

```r
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(rwalk,type='o',ylab='y')
abline(model1) # add the fitted least squares line from model1
```



pratice

```r
model1a=lm(rwalk~time(rwalk)+I(time(rwalk)^2))
summary(model1a)
```

```
##
## Call:
## lm(formula = rwalk ~ time(rwalk) + I(time(rwalk)^2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.69623 -0.76802  0.00826  0.85337  2.34468
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -1.4272911  0.4534893  -3.147  0.00262 **
## time(rwalk)       0.1746746  0.0343028   5.092 4.16e-06 ***
## I(time(rwalk)^2) -0.0006654  0.0005451  -1.221  0.22721
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.132 on 57 degrees of freedom
## Multiple R-squared:  0.8167, Adjusted R-squared:  0.8102
## F-statistic:   127 on 2 and 57 DF,  p-value: < 2.2e-16
```

```
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(rwalk,type='o',ylab='y')
abline(model1a)
```

```
## Warning in abline(model1a): only using the first two of 3 regression
## coefficients
```

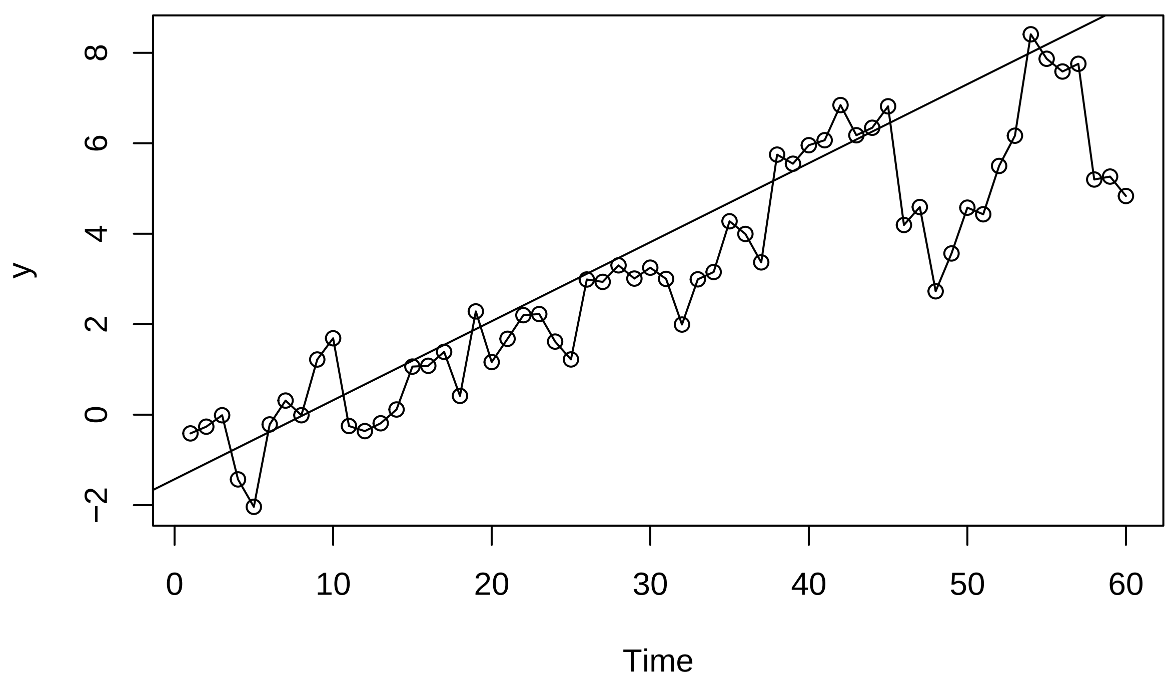


Exhibit 3.3 Regression Results for the Seasonal Means Model

```
data(tempdub)
month.=season(tempdub) # period added to improve table display
model2=lm(tempdub~month.-1) # -1 removes the intercept term
summary(model2)
```

```
##
## Call:
## lm(formula = tempdub ~ month. - 1)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2750 -2.2479  0.1125  1.8896  9.8250
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## month.January     16.608      0.987   16.83   <2e-16 ***
## month.February    20.650      0.987   20.92   <2e-16 ***
## month.March       32.475      0.987   32.90   <2e-16 ***
## month.April       46.525      0.987   47.14   <2e-16 ***
## month.May         58.092      0.987   58.86   <2e-16 ***
## month.June        67.500      0.987   68.39   <2e-16 ***
## month.July        71.717      0.987   72.66   <2e-16 ***
## month.August      69.333      0.987   70.25   <2e-16 ***
## month.September   61.025      0.987   61.83   <2e-16 ***
## month.October     50.975      0.987   51.65   <2e-16 ***
## month.November    36.650      0.987   37.13   <2e-16 ***
## month.December    23.642      0.987   23.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.419 on 132 degrees of freedom
## Multiple R-squared:  0.9957, Adjusted R-squared:  0.9953
## F-statistic:  2569 on 12 and 132 DF,  p-value: < 2.2e-16
```

pratice code

```r
sex=factor(c('M','F','M','M','F'))
sex
```

```
## [1] M F M M F
## Levels: F M
```

```r
sex=factor(c('M','F','M','M','F'),levels=c('M','F'))
sex
```

```
## [1] M F M M F
## Levels: M F
```

```r
table(sex)
```

```
## sex
## M F
## 3 2
```

```r
# fitted(model2)
# residuals(model2)
```

Exhibit 3.4 Results for Seasonal Means Model with an Intercept

```
model3=lm(tempdub~month.)
summary(model3)
```

```
##
## Call:
## lm(formula = tempdub ~ month.)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.2750 -2.2479  0.1125  1.8896  9.8250
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        16.608      0.987  16.828  < 2e-16 ***
## month.February      4.042      1.396   2.896  0.00443 **
## month.March        15.867      1.396  11.368  < 2e-16 ***
## month.April        29.917      1.396  21.434  < 2e-16 ***
## month.May          41.483      1.396  29.721  < 2e-16 ***
## month.June         50.892      1.396  36.461  < 2e-16 ***
## month.July         55.108      1.396  39.482  < 2e-16 ***
## month.August       52.725      1.396  37.775  < 2e-16 ***
## month.September    44.417      1.396  31.822  < 2e-16 ***
## month.October      34.367      1.396  24.622  < 2e-16 ***
## month.November     20.042      1.396  14.359  < 2e-16 ***
## month.December      7.033      1.396   5.039 1.51e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.419 on 132 degrees of freedom
## Multiple R-squared:  0.9712, Adjusted R-squared:  0.9688
## F-statistic: 405.1 on 11 and 132 DF,  p-value: < 2.2e-16
```

Exhibit 3.5 Cosine Trend Model for Temperature Series

```
har.=harmonic(tempdub,1)
model4=lm(tempdub~har.)
summary(model4)
```

```
##
## Call:
## lm(formula = tempdub ~ har.)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.1580  -2.2756  -0.1457   2.3754  11.2671
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       46.2660     0.3088 149.816  < 2e-16 ***
## har.cos(2*pi*t) -26.7079     0.4367 -61.154  < 2e-16 ***
## har.sin(2*pi*t)  -2.1697     0.4367  -4.968 1.93e-06 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.706 on 141 degrees of freedom
## Multiple R-squared:  0.9639, Adjusted R-squared:  0.9634
## F-statistic:  1882 on 2 and 141 DF,  p-value: < 2.2e-16
```

pratice

```
M=matrix(1:6,ncol=2)
M
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
dim(M)
```

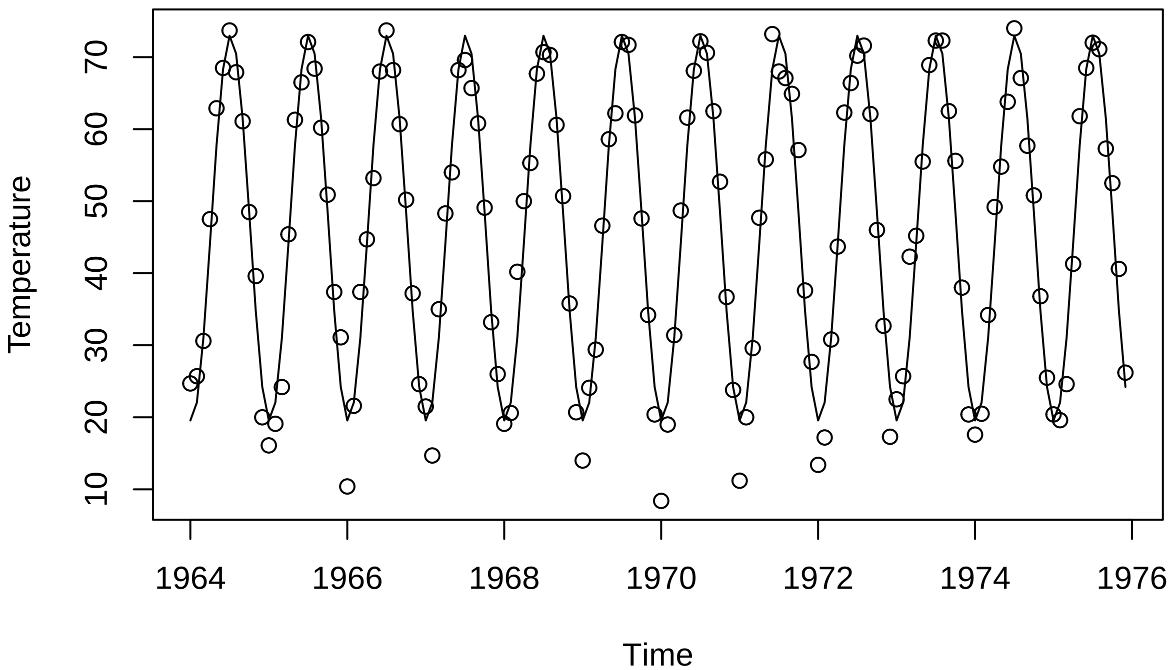
```
## [1] 3 2
```

```
apply(M, 2, mean)
```

```
## [1] 2 5
```

Exhibit 3.6 Cosine Trend for the Temperature Series

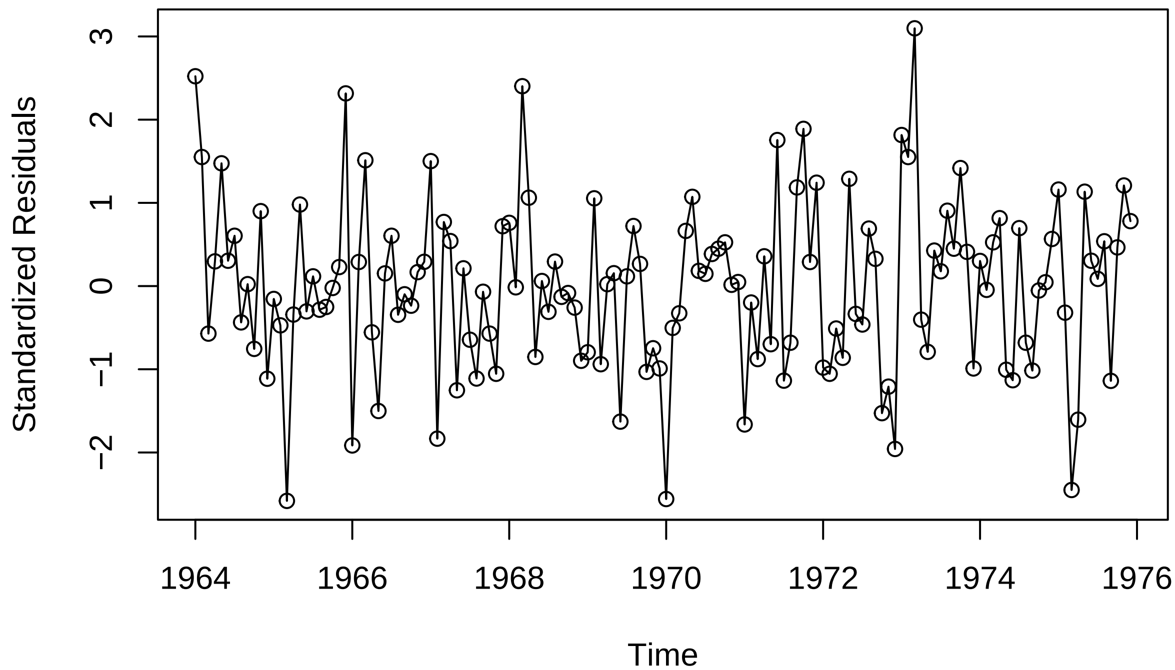
```
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(ts(fitted(model4),freq=12,start=c(1964,1)),ylab='Temperature',type='l',
ylim=range(c(fitted(model4),tempdub))); points(tempdub)
```



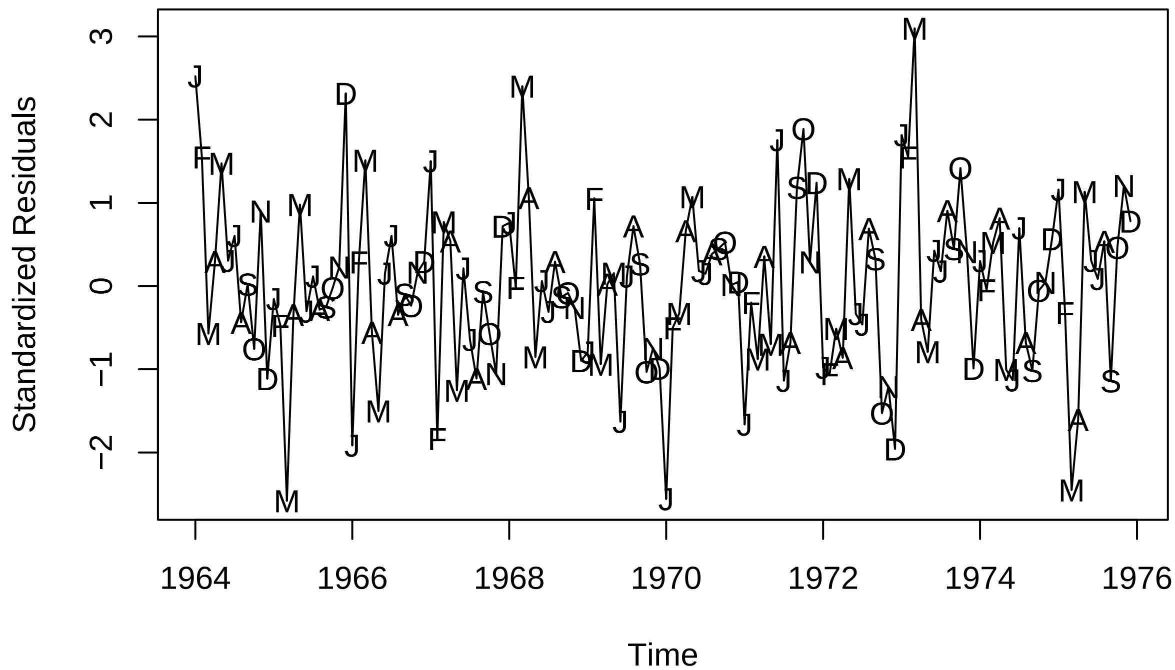

```
# ylim ensures that the y axis range fits the raw data and thefitted values
```

Exhibit 3.8 Residuals versus Time for Temperature Seasonal Means

```
# returns the (externally) Studentized residuals from the fitted mode
plot(y=rstudent(model3),x=as.vector(time(tempdub)),xlab='Time',ylab='Standardized Residuals',type='o')
```




```
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(y=rstudent(model3),x=as.vector(time(tempdub)),xlab='Time',ylab='Standardized Residuals',type='l')
points(y=rstudent(model3),x=as.vector(time(tempdub)),pch=as.vector(season(tempdub)))
```




```r
# returns the (internally) standardized residuals
plot(y=rstandard(model3),x=as.vector(time(tempdub)),xlab='Time',ylab='Standardized Residuals',type='o')
```

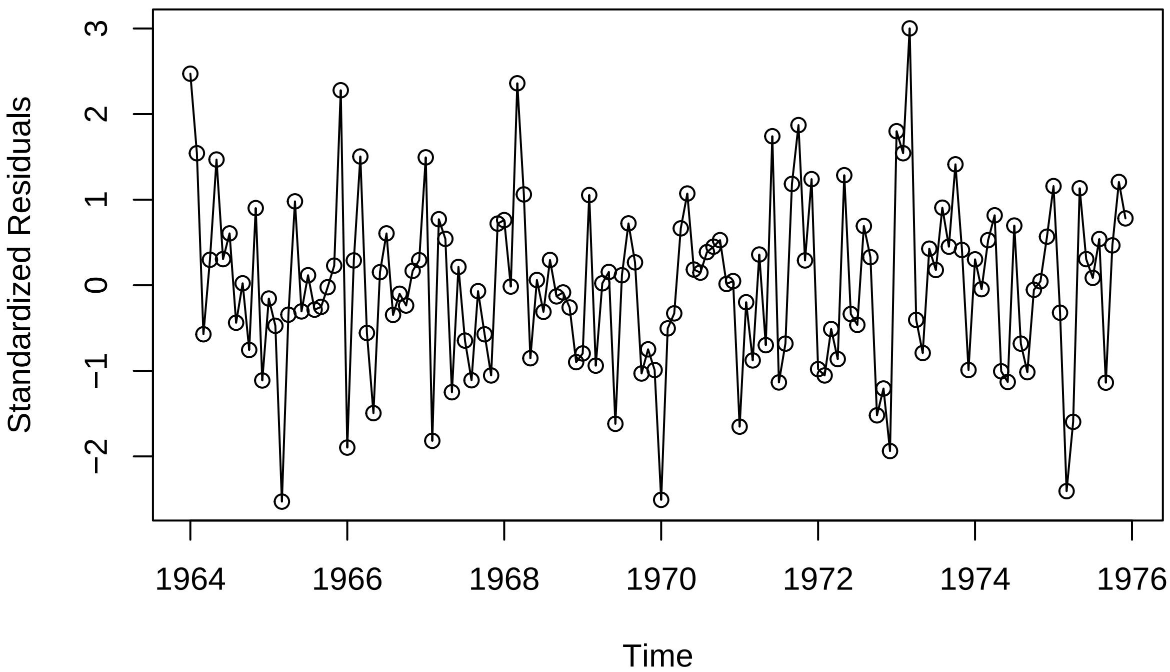


Exhibit 3.10 Standardized Residuals versus Fitted Values for the Temperature Seasonal Means Model

```
plot(y=rstudent(model3),x=as.vector(fitted(model3)),xlab='Fitted Trend Values', ylab='Standardized Resi
points(y=rstudent(model3),x=as.vector(fitted(model3)),pch=as.vector(season(tempdub)))
```

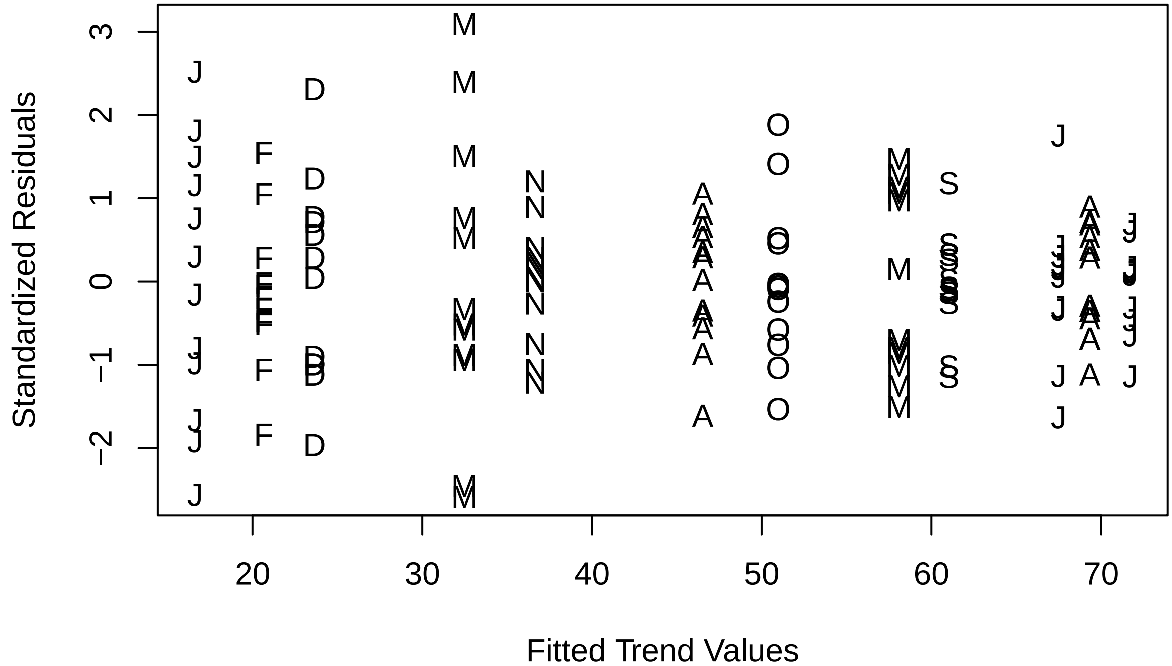


Exhibit 3.11 Histogram of Standardized Residuals from Seasonal Means Model

```
hist(rstudent(model3),xlab='Standardized Residuals', main='Histogram of the Standardized Residuals')
```

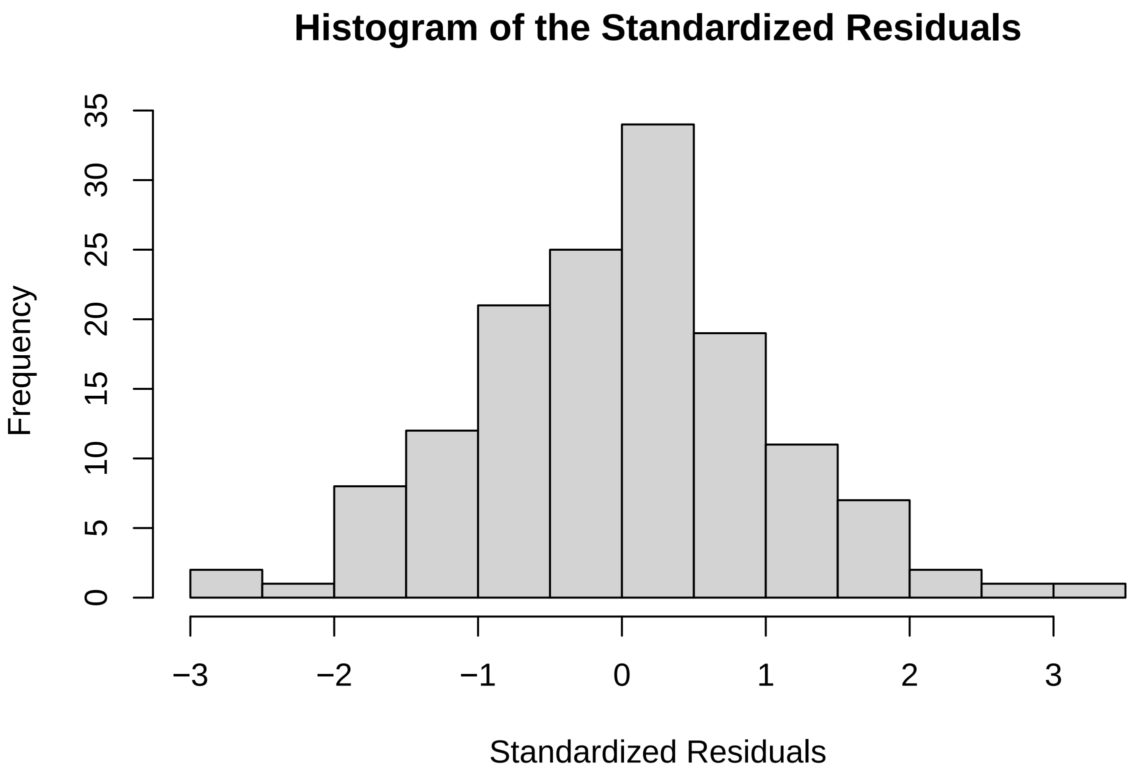


Exhibit 3.12 Q-Q Plot: Standardized Residuals of Seasonal Means Model

```
qqnorm(rstudent(model3))
qqline(rstudent(model3))   # draws a line
```

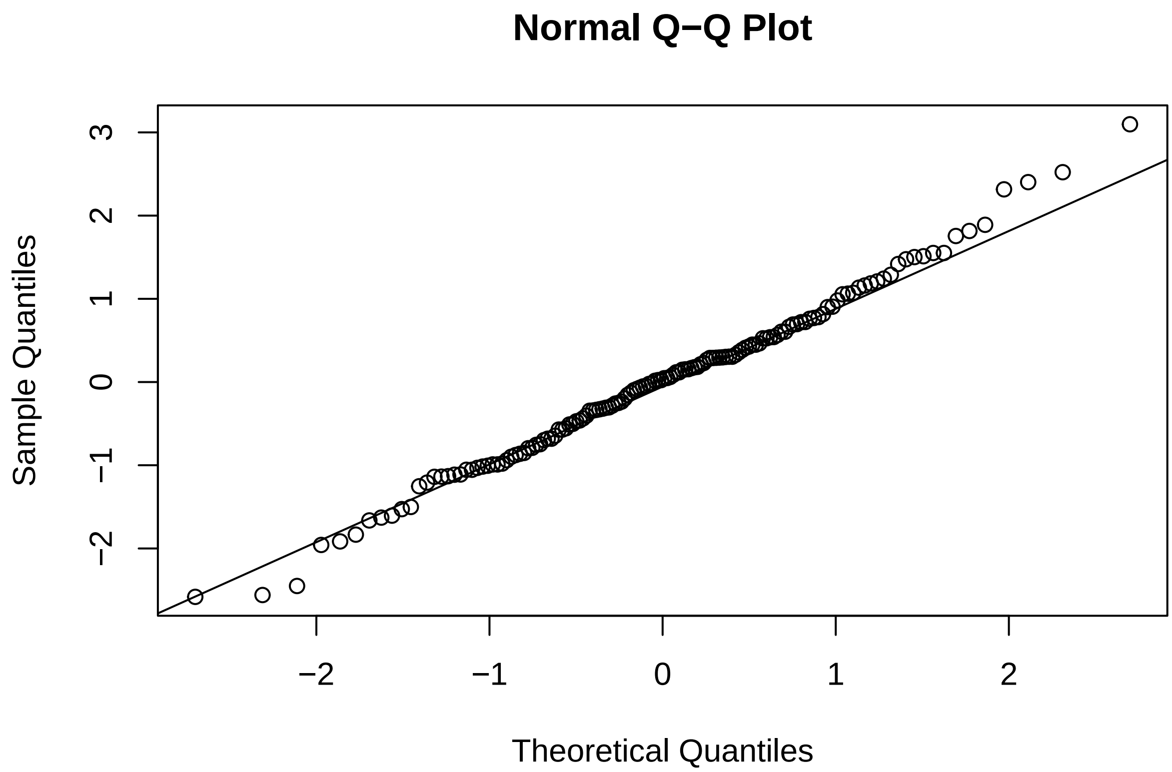
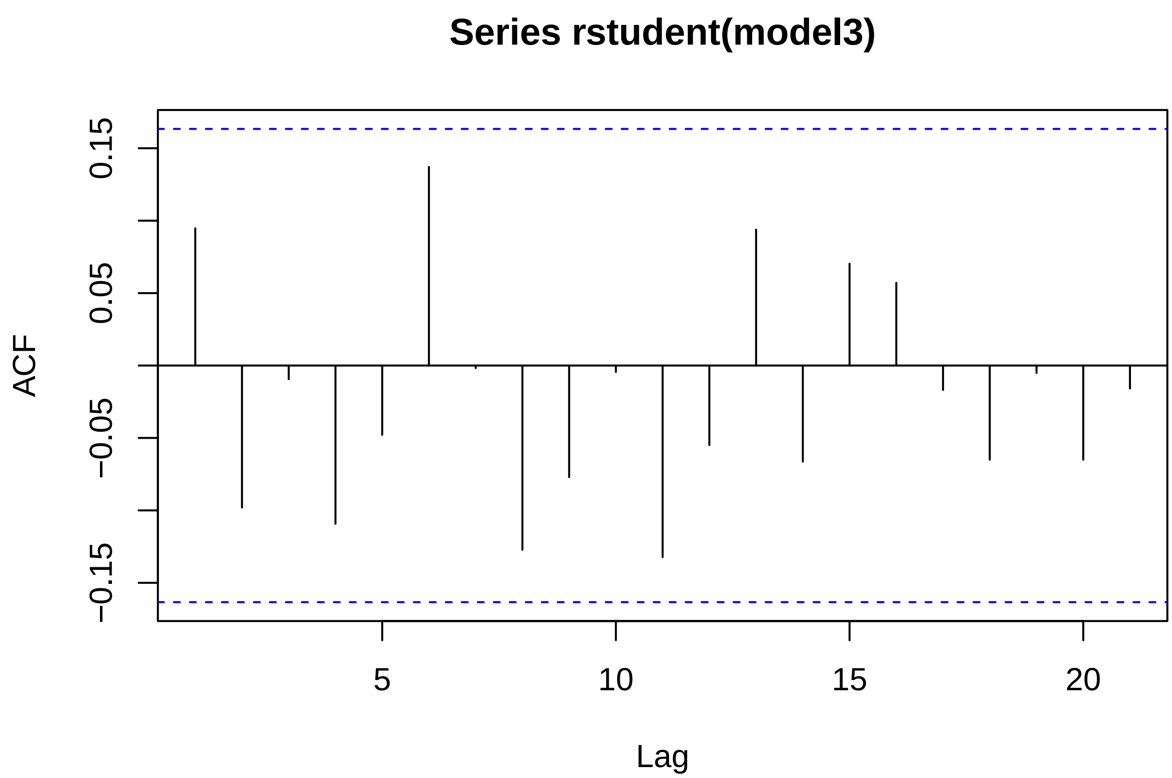


Exhibit 3.13 Sample Autocorrelation of Residuals of Seasonal MeansModel

```r
acf(rstudent(model3)) # computes the sample autocorrelation function of the time series
```




```
shapiro.test(rstudent(model3))
```

```
##
##  Shapiro-Wilk normality test
##
## data:  rstudent(model3)
## W = 0.9929, p-value = 0.6954
```

```
runs(rstudent(model3))
```

```
## $pvalue
## [1] 0.216
##
## $observed.runs
## [1] 65
##
## $expected.runs
## [1] 72.875
##
## $n1
## [1] 69
##
## $n2
## [1] 75
##
```

```
## $k
## [1] 0
```

Exhibit 3.14 Residuals from Straight Line Fit of the Random Walk

```
plot(y=rstudent(model1),x=as.vector(time(rwalk)),ylab='Standardized Residuals',xlab='Time',type='o')
```

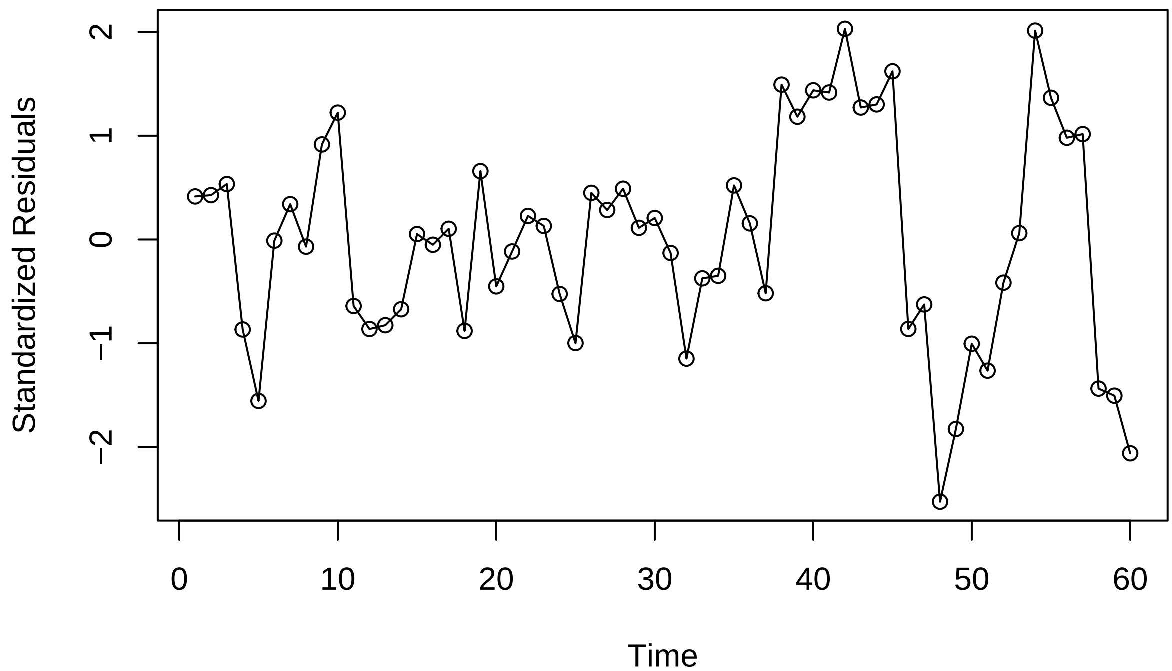


Exhibit 3.15 Residuals versus Fitted Values from Straight Line Fit

```
plot(y=rstudent(model1),x=fitted(model1),
ylab='Standardized Residuals',xlab='Fitted Trend Line Values',
type='p')
```

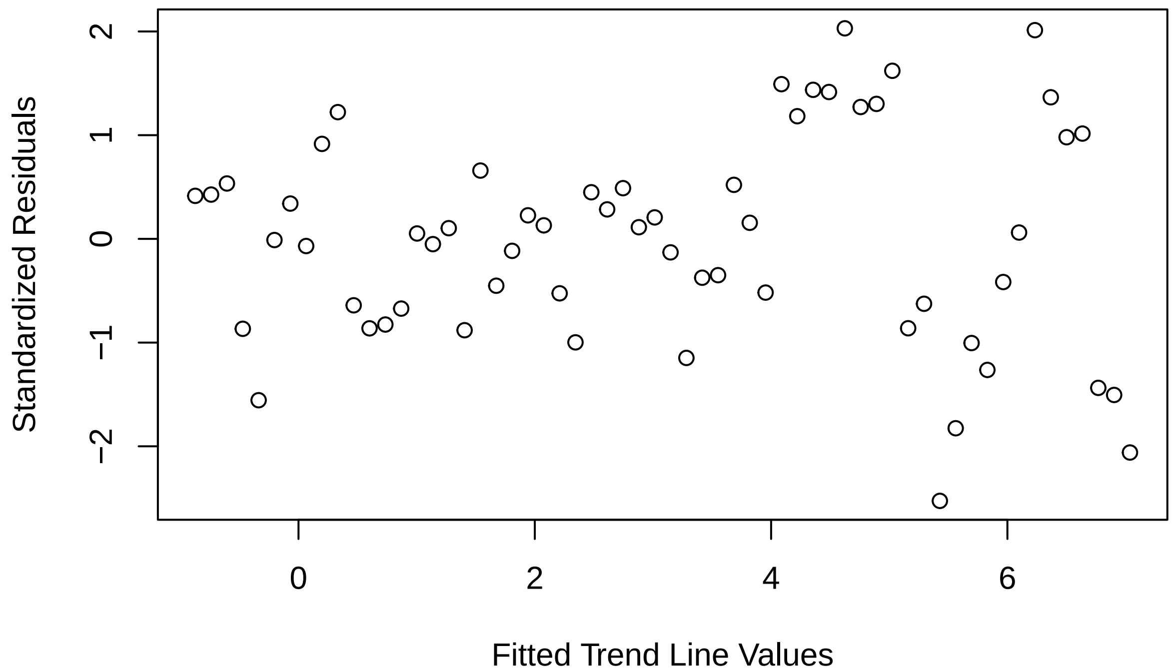


Exhibit 3.16 Sample Autocorrelation of Residuals from Straight Line Model

```
acf(rstudent(model1))
```

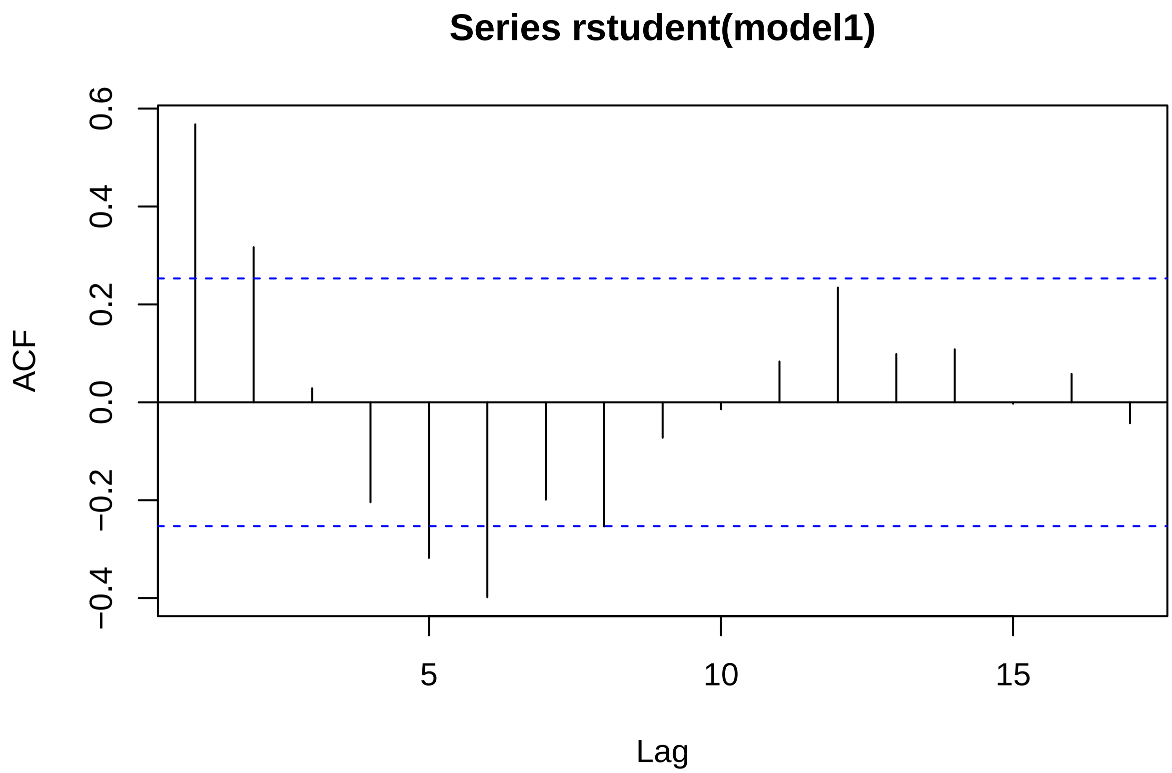
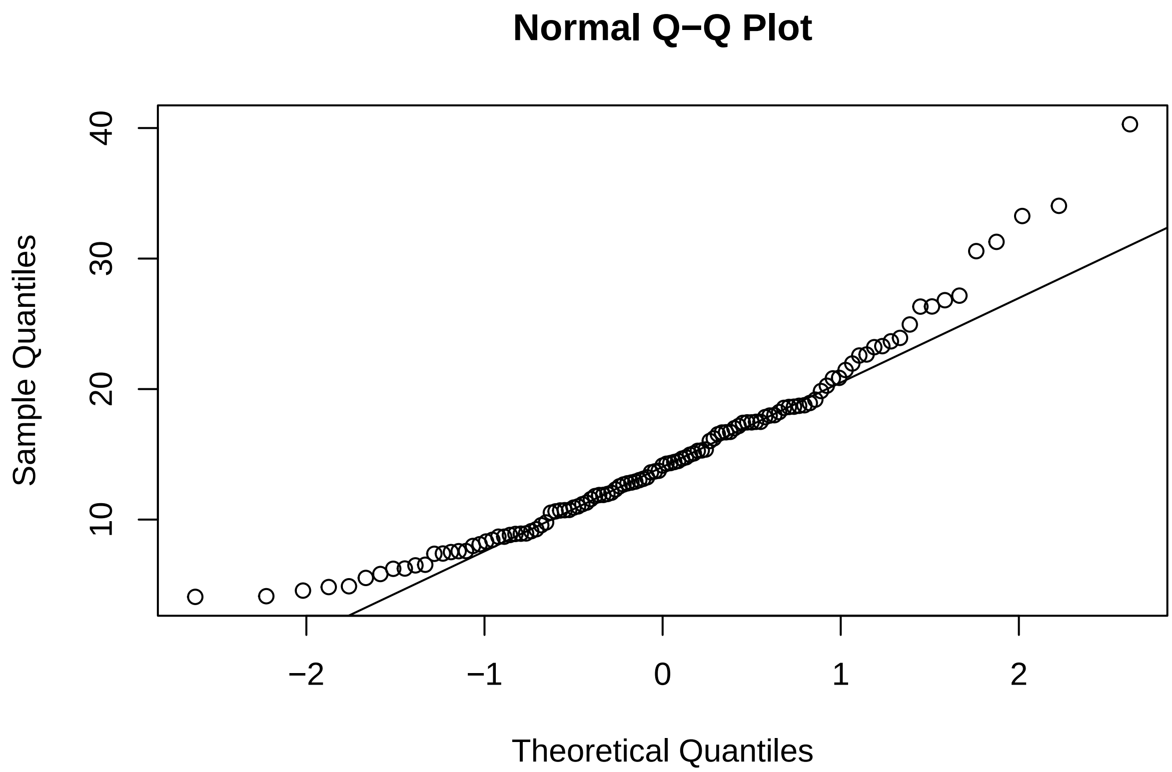


Exhibit 3.17 Quantile-Quantile Plot of Los Angeles Annual Rainfall Series

```r
qqnorm(larain); qqline(larain)
```



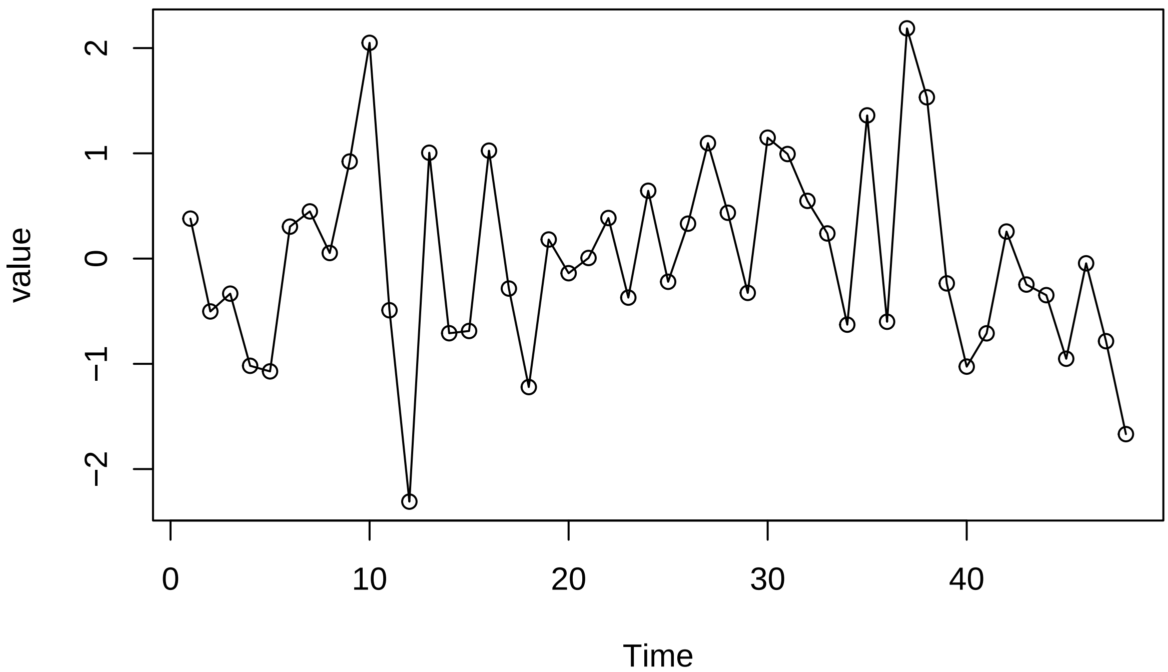
Question 2:

1.3 : Simulate a completely random process of length 48 with independent, normal values. Plot the time series plot. Does it look "random"? Repeat this exercise several times with a new simulation each time.

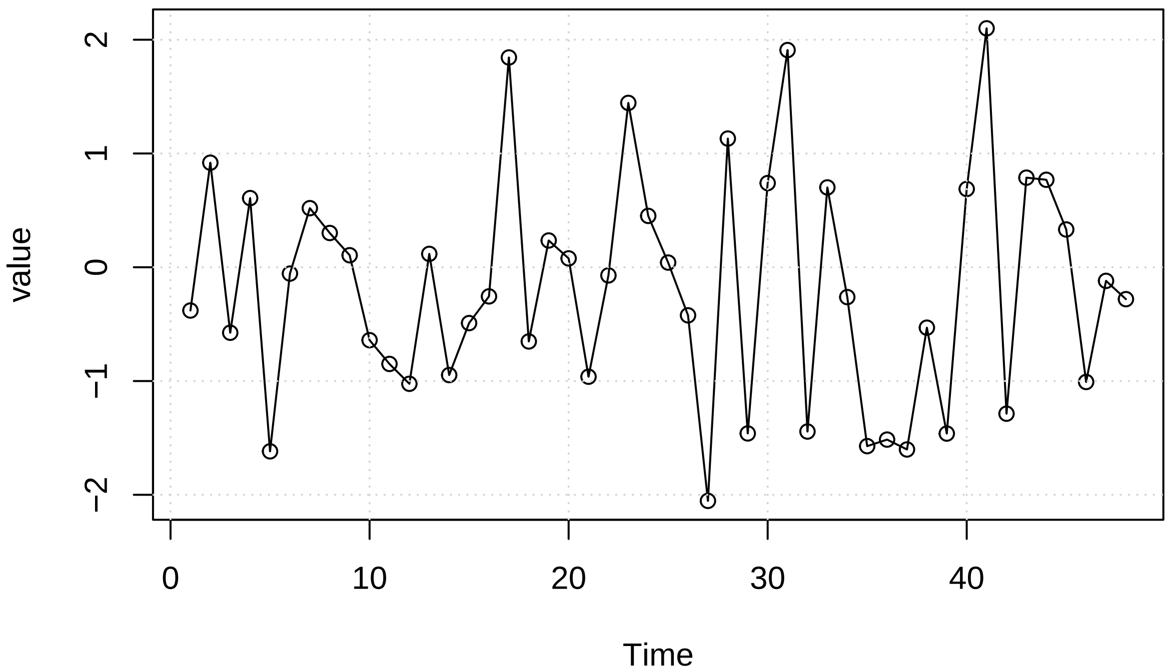
```
n=48
random_values=rnorm(n, mean = 0, sd = 1)
head(random_values)
```

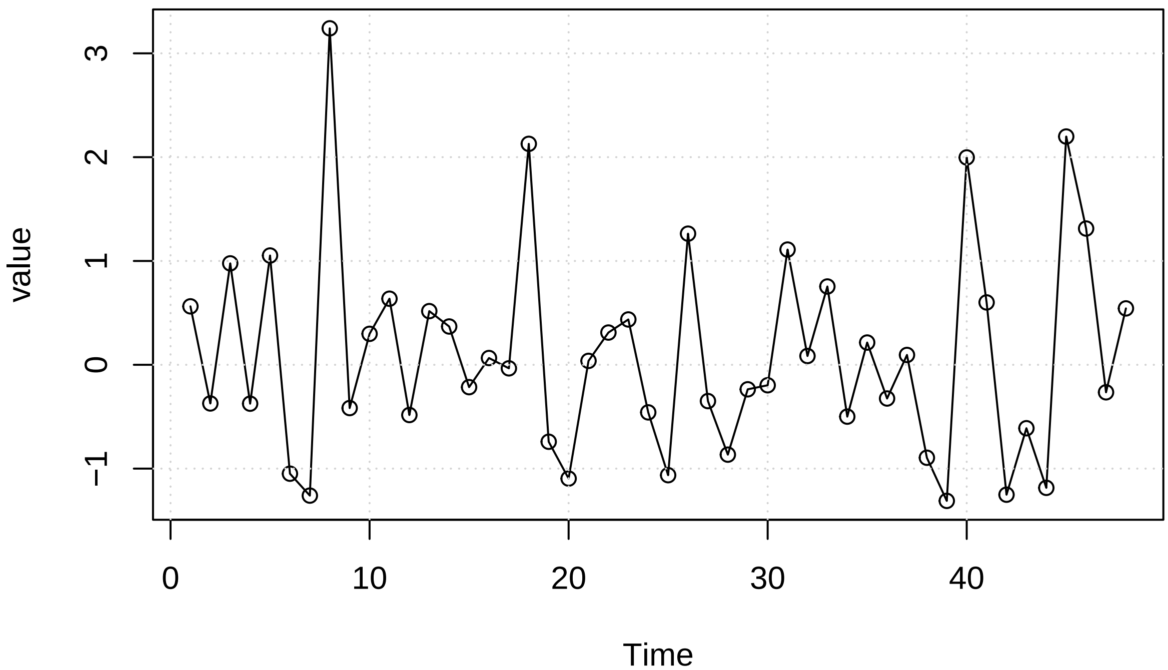
```
## [1]  0.3796395 -0.5023235 -0.3332074 -1.0185754 -1.0717912  0.3035286
```

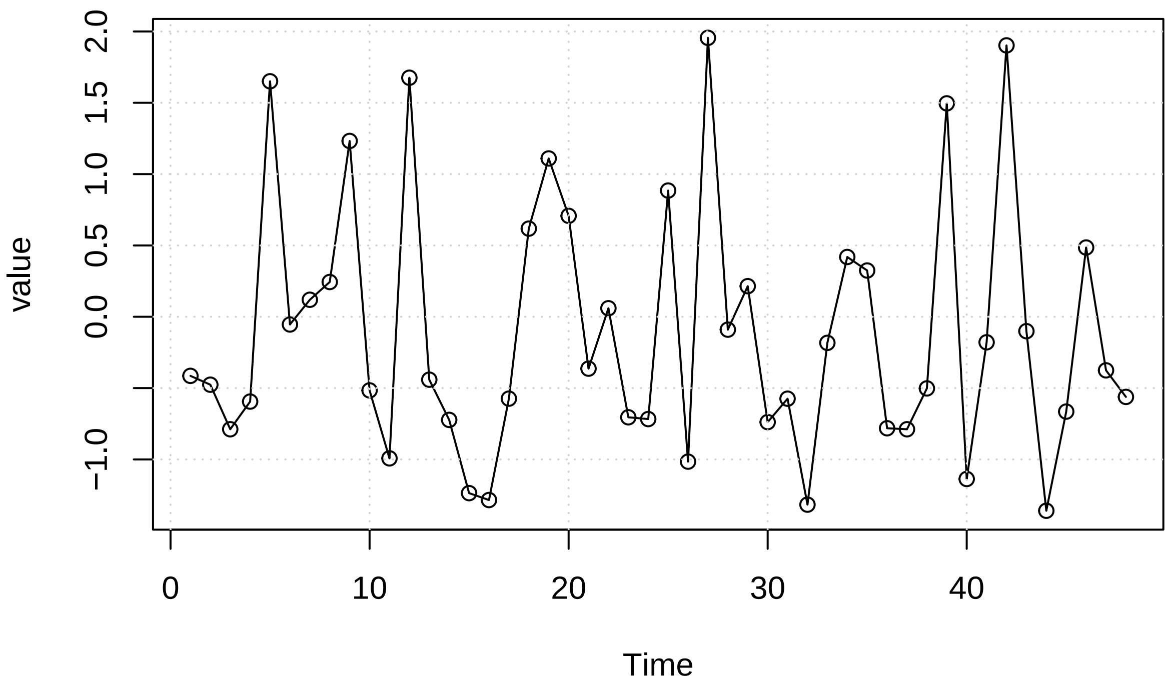
```
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(ts(random_values), xlab = 'Time', ylab='value', type='o')
```




```
num_simulations = 3
for (i in 1:num_simulations) {
    random_values=rnorm(n, mean = 0, sd = 1)
    # win.graph(width=4.875, height=2.5,pointsize=8)
    plot(ts(random_values), xlab = 'Time', ylab='value', type='o')
    grid()
}
```


value
Time
2
1
0
−1
−2
0
10
20
30
40

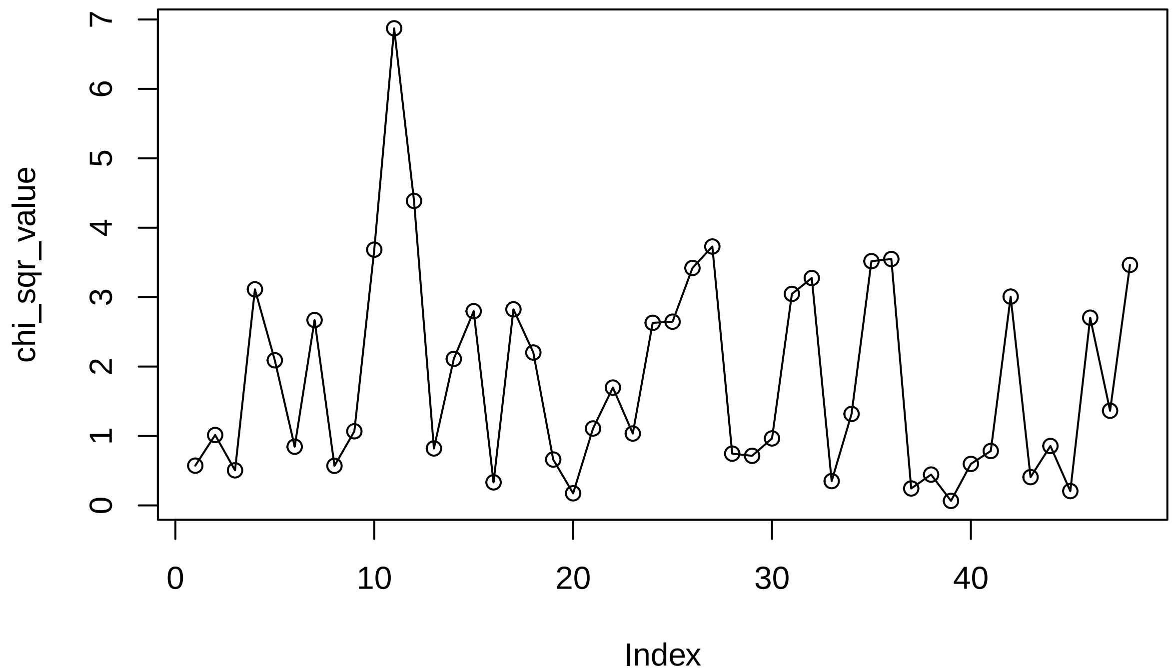

value
3
2
1
0
−1
0
10
20
30
40
Time



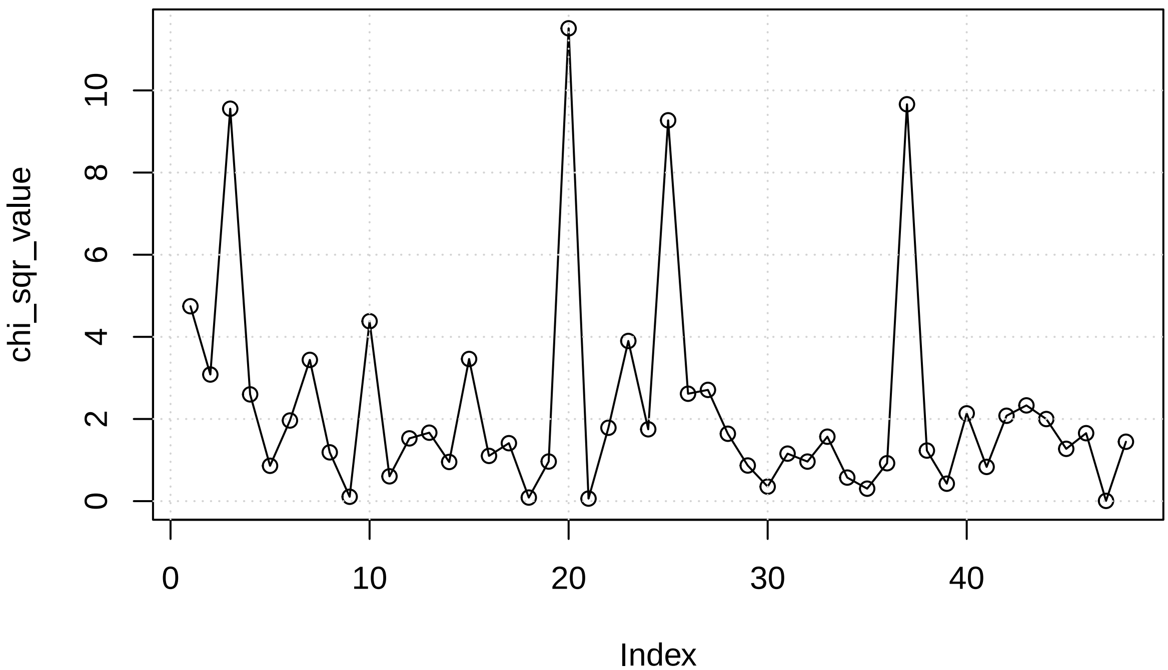
Observation : After executing the script, I observed three time series plots, each illustrating a distinct simulation of the random process. The plots appear "random," displaying fluctuations without any noticeable trend or pattern. This behavior is typical of a stochastic process generated from a normal distribution. Each time I run the simulation, I see different plots because of the inherent randomness of the values.

1.4: Simulate a completely random process of length 48 with independent, chi-square distributed values, each with 2 degrees of freedom. Display the time series plot. Does it look "random" and nonnormal? Repeat this exercise several times with a new simulation each time.

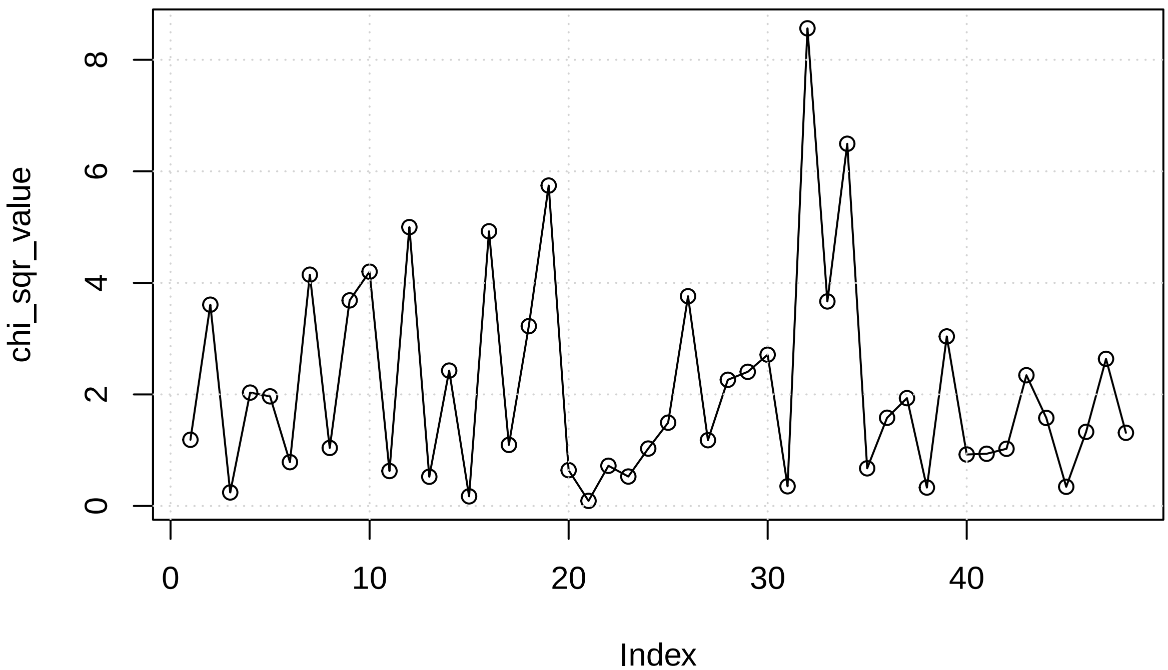
```
chi_sqr_value = rchisq(n, df = 2)
# win.graph(width=4.875, height=2.5,pointsize=8)
plot(chi_sqr_value,type = 'o')
```

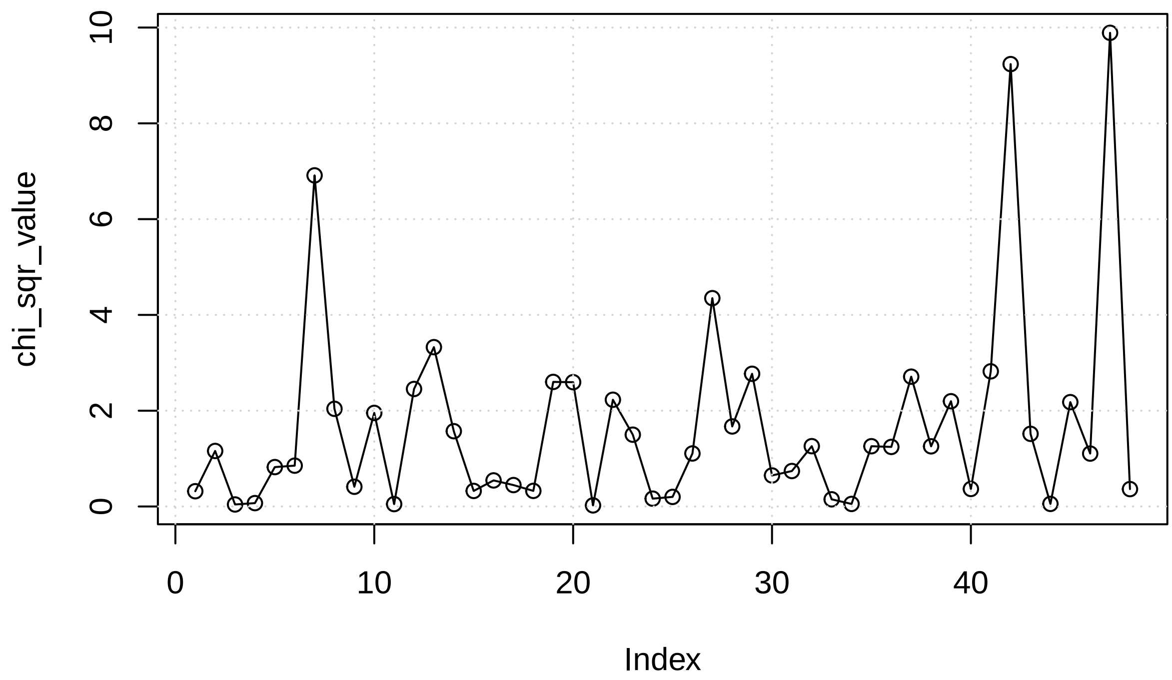



```
num = 3
for (i in 1:num)
{
    chi_sqr_value = rchisq(n, df = 2)
    # win.graph(width=4.875, height=2.5,pointsize=8)
    plot(chi_sqr_value,type = 'o')
    grid()
}
```



chi_sqr_value
0
2
4
6
8
10
0
10
20
30
40
Index

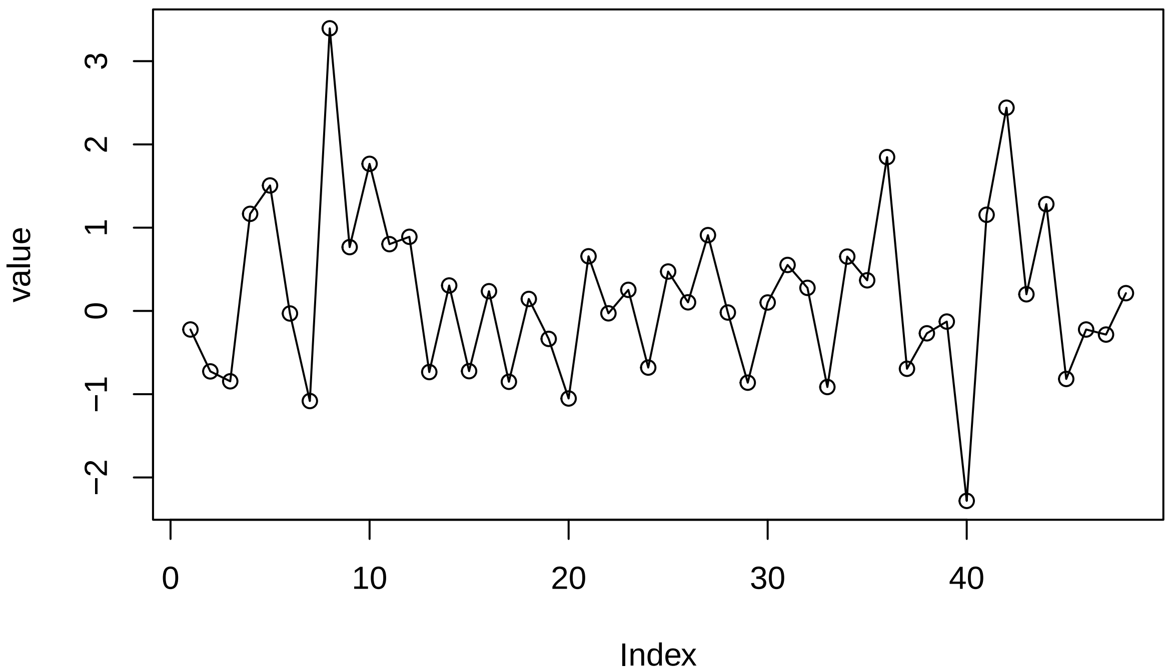

chi_sqr_value
0
2
4
6
8
0
10
20
30
40
Index



Observation: After executing the script, I observed three time series plots, each depicting a distinct simulation of the random process. The plots appear "random," displaying fluctuations; however, they do not conform to a normal distribution shape. Values drawn from a chi-square distribution, particularly with 2 degrees of freedom, typically exhibit positive skewness. Each time I run the simulation, I encounter different plots because of the inherent randomness of the values.

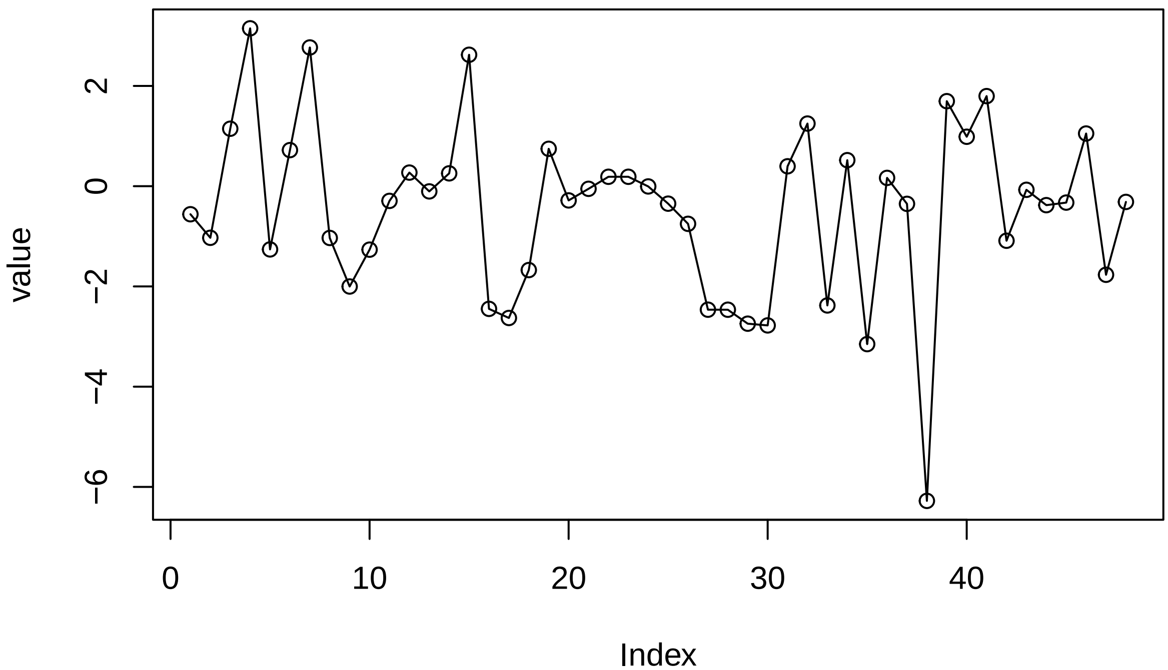
1.5 : Simulate a completely random process of length 48 with independent, t-distributed values each with 5 degrees of freedom. Construct the time series plot. Does it look "random" and nonnormal? Repeat this exercise several times with a new simulation each time

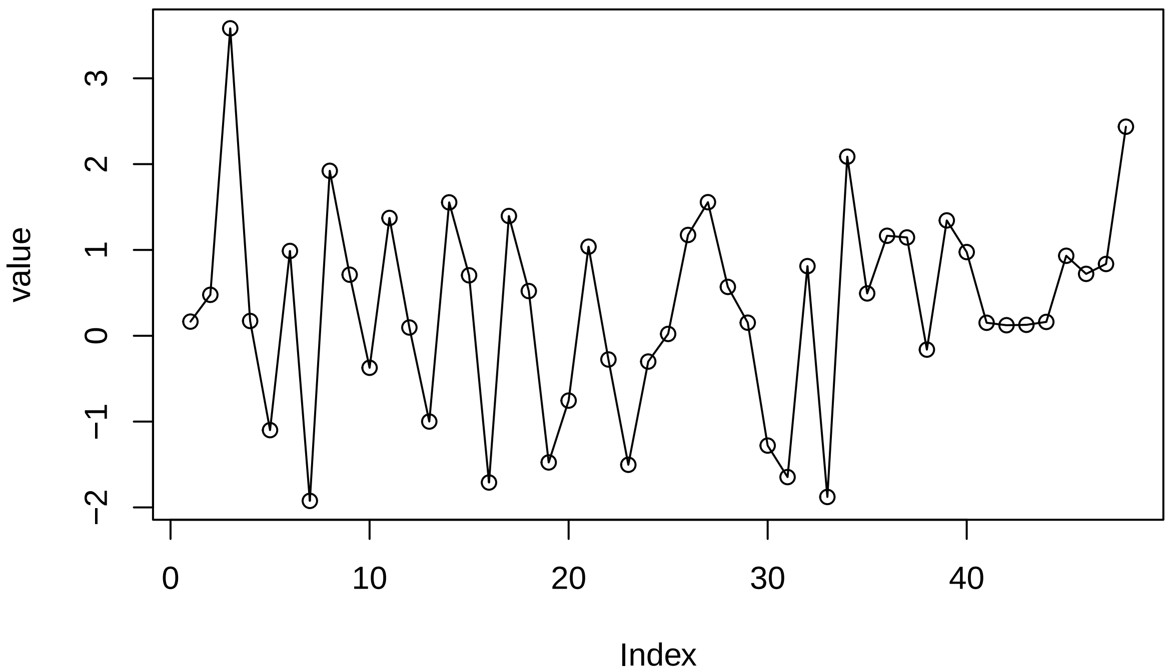
```
n = 48
t_dis_value =  rt(n, df = 5)
plot(t_dis_value, type = 'o', ylab='value')
```



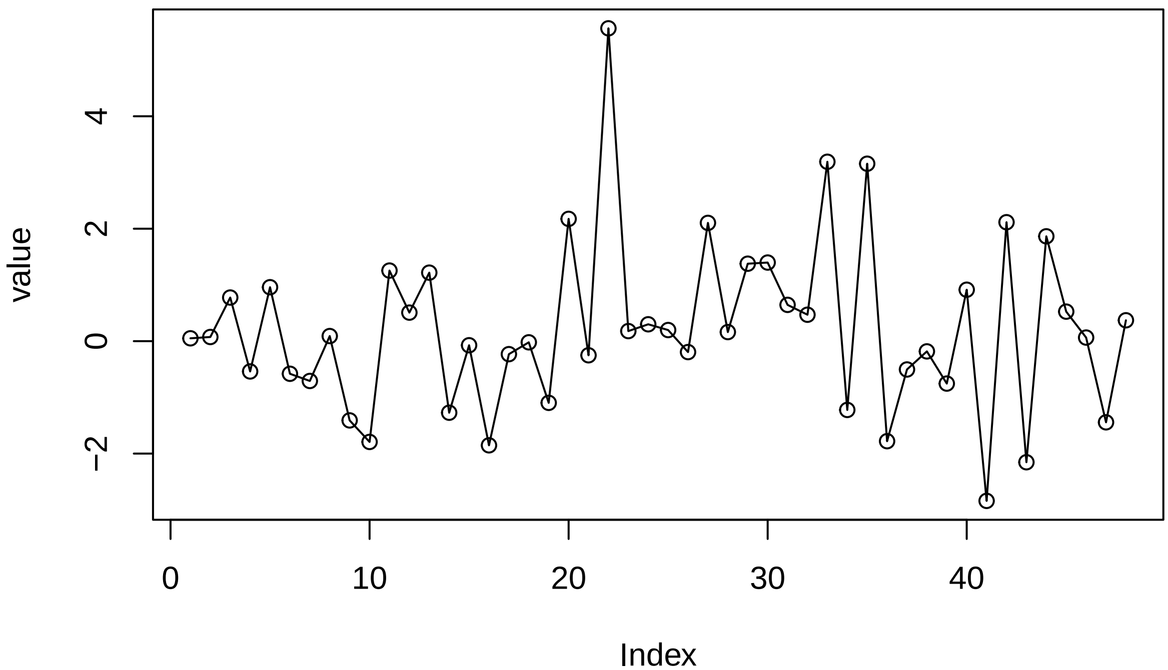
simulation

```
nums = 3
for (i in 1:nums)
{
    t_dis_value =  rt(n, df = 5)
    plot(t_dis_value, type = 'o', ylab='value')
}
```


value
2
0
−2
−4
−6
0
10
20
30
40
Index



value
3
2
1
0
−1
−2
0
10
20
30
40
Index



Observation : After executing the script, I observed three time series plots, each illustrating a different simulation of the random process. The plots appear "random," displaying fluctuations. However, since the values are drawn from a t-distribution with 5 degrees of freedom, they may exhibit heavier tails compared to a normal distribution, resulting in a nonnormal appearance. Each time I run the simulation, I see different plots due to the inherent randomness of the values.