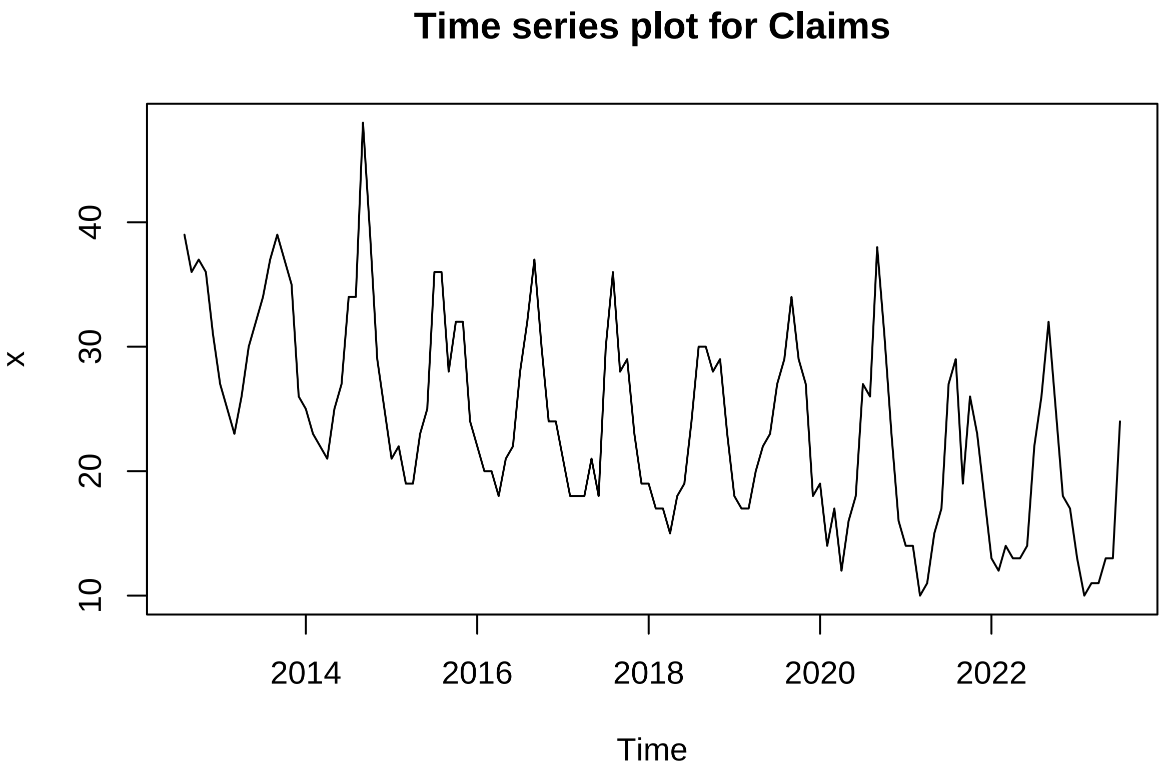
a)

```
plot(Claims, main = "Time series plot for Claims")
```

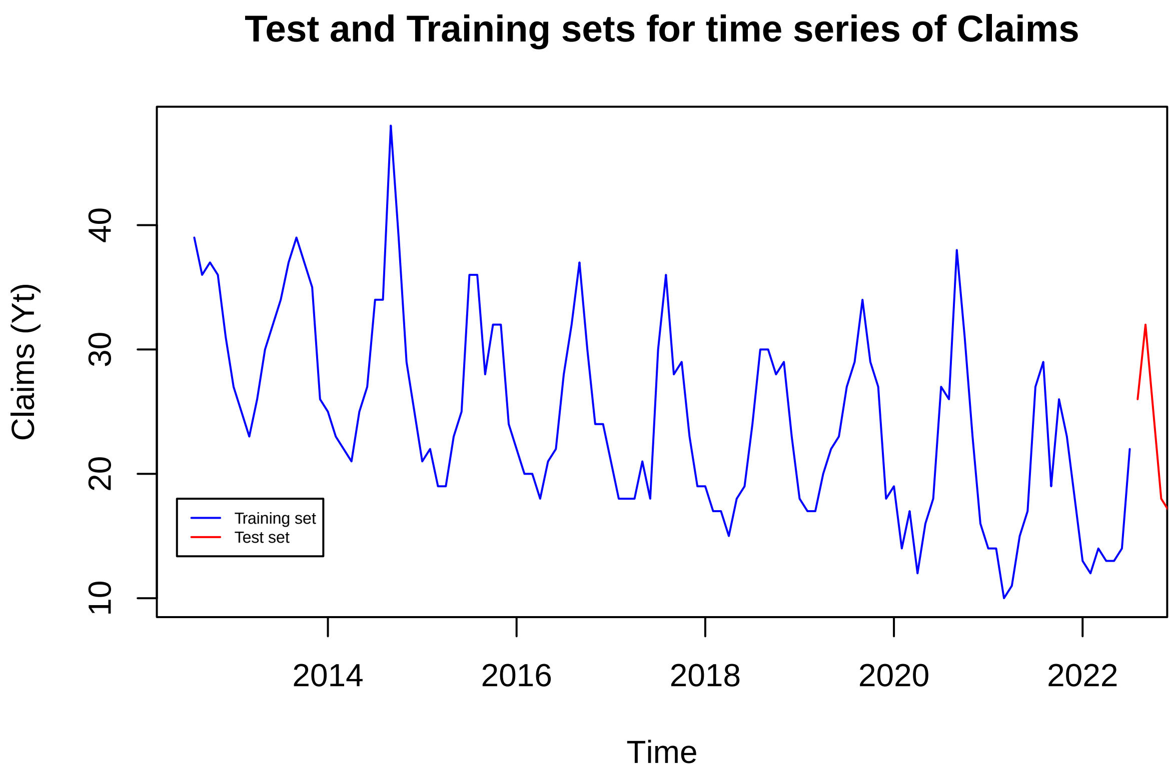


Comment:

- In the time series plot there seems to be seasonality. The period seems to be constant as a function of time which suggests the covariance among all claim observations is constant.
- The mean of the time series seems to be decreasing slightly with time. One could draw a line through the midpoints of the Y_t midpoints for some intervals of t to see this slightly negative slope.
- The variance looks like it could be constant with respect to time. However there are three peaks that stand out, suggesting the variance for these observations is unusually large.

b)

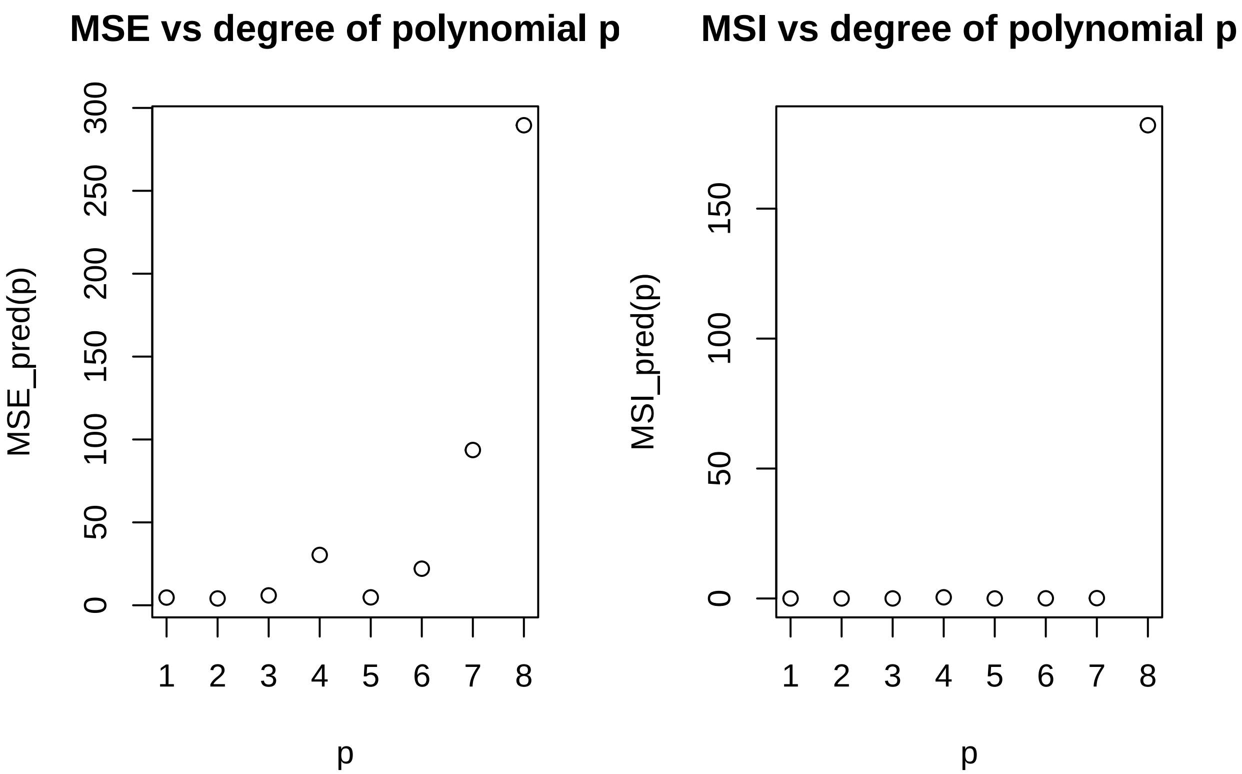
```
training_set <- window(Claims, start=start(Claims), end=time(Claims)[120])
test_set <-window(Claims, start=time(Claims)[121], end=end(Claims))
plot(training_set, col="blue", ylab = "Claims (Yt)", main = "Test and Training sets for time series of C
lines(test_set, col="red")
legend(2012.4, 18, legend = c("Training set","Test set"), col=c("blue", "red"), lty=1:1, cex=.5)
```



c)

```
tim <- time(training_set)
month <-as.factor(cycle(training_set))

reg_p1 <- lm(training_set ~ poly(as.vector(tim), 1) + month) # Orthogonal polynomial deg 1
reg_p2 <- lm(training_set ~ poly(as.vector(tim), 2) + month) # Orthogonal polynomial deg 2
reg_p3 <- lm(training_set ~ poly(as.vector(tim), 3) + month) # Orthogonal polynomial deg 3
reg_p4 <- lm(training_set ~ poly(as.vector(tim), 4) + month) # Orthogonal polynomial deg 4
reg_p5 <- lm(training_set ~ poly(as.vector(tim), 5) + month) # Orthogonal polynomial deg 5
reg_p6 <- lm(training_set ~ poly(as.vector(tim), 6) + month) # Orthogonal polynomial deg 6
reg_p7 <- lm(training_set ~ poly(as.vector(tim), 7) + month) # Orthogonal polynomial deg 7
reg_p8 <- lm(training_set ~ poly(as.vector(tim), 8) + month) # Orthogonal polynomial deg 8


tim.new <- as.vector(time(test_set))
month.new  <-as.factor(cycle(test_set))
new <- data.frame(tim=tim.new,month=month.new)


pred_p1 <- predict.lm(reg_p1, new)
pred_p2 <- predict.lm(reg_p2, new)
pred_p3 <- predict.lm(reg_p3, new)
pred_p4 <- predict.lm(reg_p4, new)
pred_p5 <- predict.lm(reg_p5, new)
pred_p6 <- predict.lm(reg_p6, new)
pred_p7 <- predict.lm(reg_p7, new)
pred_p8 <- predict.lm(reg_p8, new)

# takes in a vector of ordered predictions
Y_test = as.vector(test_set)

mse_pred = function(Y_pred){
  mean((Y_test - Y_pred)^2)
}

msi_pred = function(Y_pred){
  mean((1-Y_test/Y_pred)^2)
}

p = seq(1,8)
MSEs = c(mse_pred(pred_p1),
         mse_pred(pred_p2),
         mse_pred(pred_p3),
         mse_pred(pred_p4),
         mse_pred(pred_p5),
         mse_pred(pred_p6),
         mse_pred(pred_p7),
         mse_pred(pred_p8))

MSIs = c(msi_pred(pred_p1),
         msi_pred(pred_p2),
         msi_pred(pred_p3),
```

```
         msi_pred(pred_p4),
         msi_pred(pred_p5),
         msi_pred(pred_p6),
         msi_pred(pred_p7),
         msi_pred(pred_p8))

par(mfrow=c(1,2))
plot(p, MSEs, ylab = "MSE_pred(p)",
     main = "MSE vs degree of polynomial p")
plot(p, MSIs, ylab = "MSI_pred(p)",
     main = "MSI vs degree of polynomial p")
```



Minimum value for MSEs is obtained on the second index, so the MSE divergence criteria picks p=2

```
min(MSEs)
```

```
## [1] 4.110931
```

```
MSEs
```

```
## [1]   4.646813   4.110931   5.946326  30.354078   4.772936  22.063624  93.645240
## [8] 289.542257
```

Minimum value for MSIs is obtained on the second index, so the MSI divergence criteria picks p=2

```
min(MSIs)
```

```
## [1] 0.009876646
```

```
MSIs
```

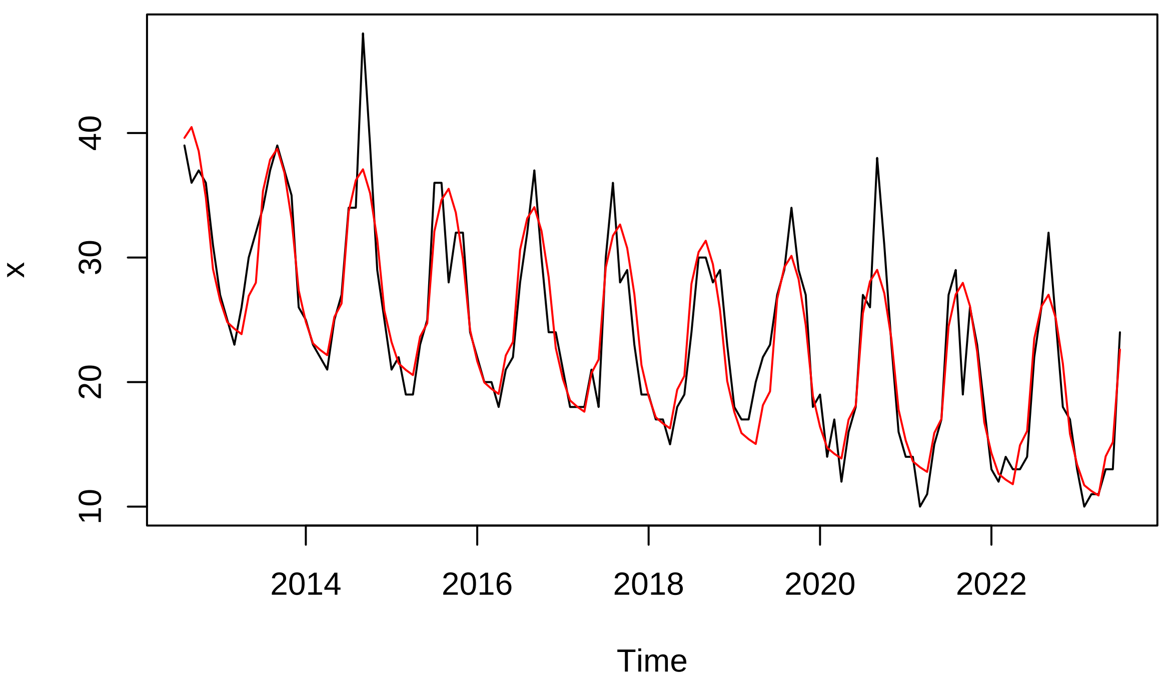
```
## [1] 1.093456e-02 9.876646e-03 1.790474e-02 4.458357e-01 1.104719e-02
## [6] 5.454036e-02 1.333643e-01 1.820798e+02
```

Both loss functions picked the same model with polynomial of degree two.

```r
# using reg_p2 (regression polynomial degree 2)
reg_p2
```

```
##
## Call:
## lm(formula = training_set ~ poly(as.vector(tim), 2) + month)
##
## Coefficients:
##             (Intercept)  poly(as.vector(tim), 2)1  poly(as.vector(tim), 2)2
##                 19.8470                  -42.9129                    3.5604
##                  month2                    month3                    month4
##                 -1.5869                   -1.9744                   -2.2625
##                  month5                    month6                    month7
##                  0.9487                    2.1594                    9.6694
##                  month8                    month9                   month10
##                 12.3255                   13.3416                   11.5571
##                 month11                   month12
##                  7.9720                    2.3863
```

```r
# fitting the reg_p2 model to the Claims time
tim <- time(Claims)
t <- as.vector(tim)
t_ <- data.frame(tim=t,month=as.factor(cycle(Claims)))
mymodel <- predict.lm(reg_p2, t_)

plot(Claims)
# plotting predictions for reg_p2
lines(mymodel ~ t,type='l',col='red')
```

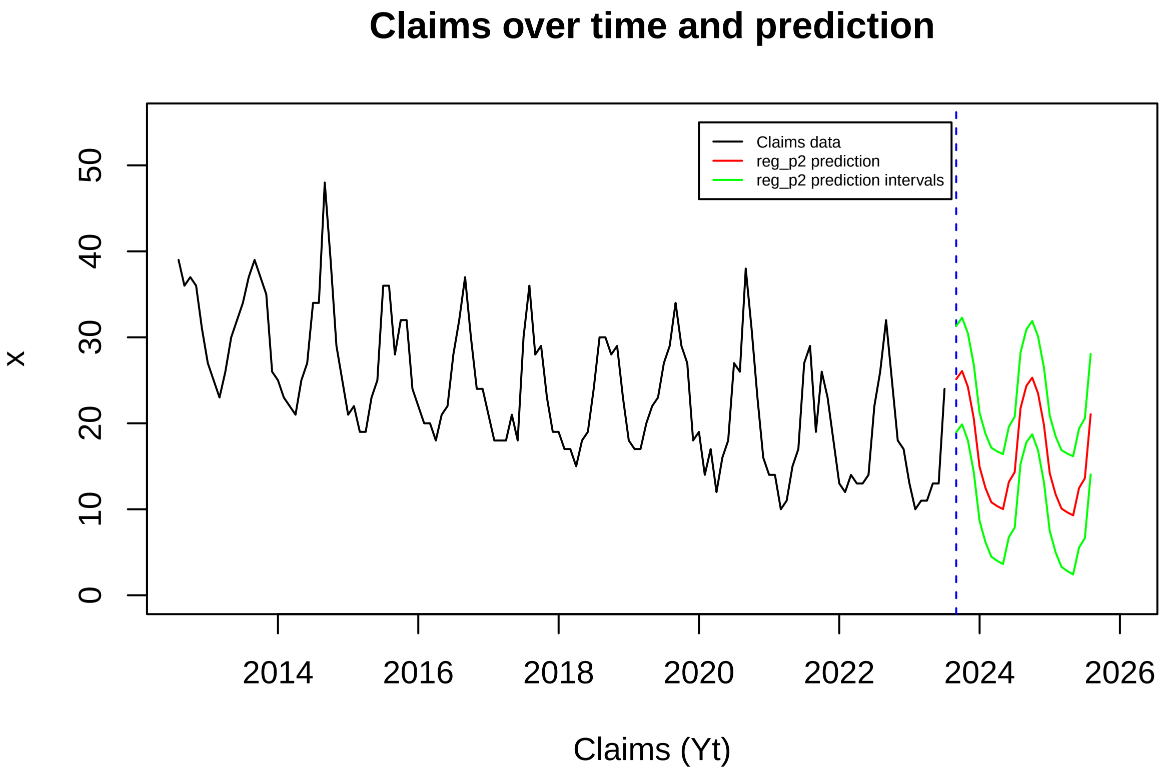


d)

```
tim.new <- seq(2023+8/12, 2025+7/12, by=1/12)
mons<- c(8:12,1:12, 1:7)
month.new <- factor(mons)
new <- data.frame(tim=tim.new,month=month.new)
# Forecast for the next 2 years using the chosen model reg_p2:
a <- predict.lm(reg_p2,new,interval='prediction', level=0.95)
# Numerical values for predictions along with prediction intervals:
a
```

```
##          fit       lwr      upr
## 1  25.133154 18.951175 31.31513
## 2  26.081007 19.875355 32.28666
## 3  24.228861 17.998752 30.45897
## 4  20.576714 14.321376 26.83205
## 5  14.924567  8.643236 21.20590
## 6  12.472420  6.164342 18.78050
## 7  10.820274  4.484702 17.15584
## 8  10.368127  4.004328 16.73193
## 9  10.015980  3.623228 16.40873
## 10 13.163834  6.741412 19.58625
## 11 14.311687  7.858891 20.76448
## 12 21.759540 15.275673 28.24341
## 13 24.353957 17.802957 30.90496
## 14 25.309079 18.720791 31.89737
## 15 23.464201 16.837745 30.09066
## 16 19.819323 13.153837 26.48481
## 17 14.174446  7.469080 20.87981
## 18 11.729568  4.983490 18.47565
## 19 10.084690  3.297082 16.87230
## 20  9.639812  2.809870 16.46975
## 21  9.294934  2.421870 16.16800
## 22 12.450056  5.533097 19.36702
## 23 13.605178  6.643564 20.56679
## 24 21.060300 14.053286 28.06731
```

```
plot(Claims,xlim=c(2012+8/12,2026), ylim=c(0, 55),
     xlab = "Claims (Yt)", main="Claims over time and prediction")
abline(v=2023+8/12,col='blue',lty=2) # adding a vertical line at the point where prediction starts
lines(a[,1]~tim.new,type='l',col='red')# plotting the predict
lines(a[,2]~tim.new,col='green') # plotting lower limit of the prediction interval
lines(a[,3]~tim.new,col='green') # plotting upper limit of the  prediction interval
legend(2020, 55, legend = c("Claims data","reg_p2 prediction", "reg_p2 prediction intervals"), col=c("bl
```



Claims over time and prediction
Claims data
reg_p2 prediction
reg_p2 prediction intervals
x
0
10
20
30
40
50
2014
2016
2018
2020
2022
2024
2026
Claims (Yt)

e) Residual Diagnostics

We must test residuals for normality, zero mean, constant variance, and randomness (iid). Note since we used the training set to train the model, we will use only the residuals of the training set.

Normality test

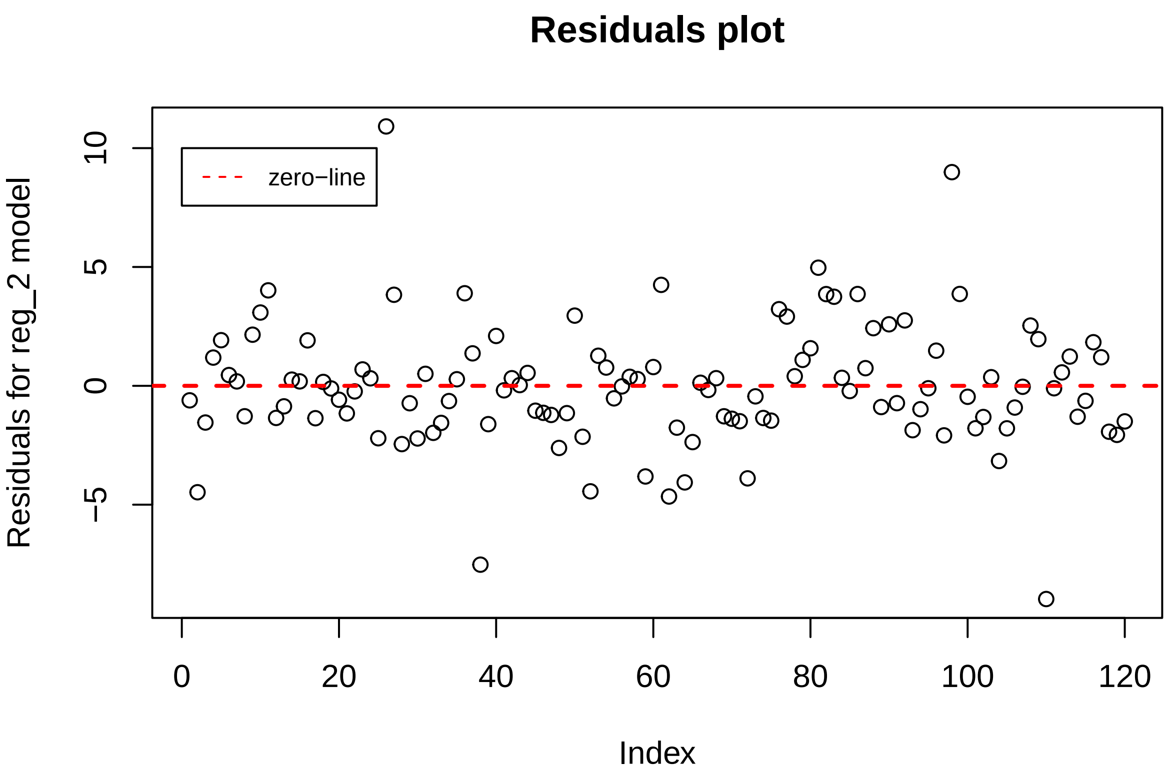
```
# Model used
mymodel <- reg_p2
# Testing normality with Shapiro-Wilk
residuals.reg = residuals(mymodel) #extracting the residuals
shapiro.test(residuals.reg) #Shapiro-Wilk test of normality H0: residuals are normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  residuals.reg
## W = 0.93537, p-value = 2.114e-05
```

The Null hypothesis for the Shapiro-Wilk test is that the residuals are normally distributed. the small $p-$value suggests there is strong evidence against this. So this assumption is **not satisfied**

Zero-mean

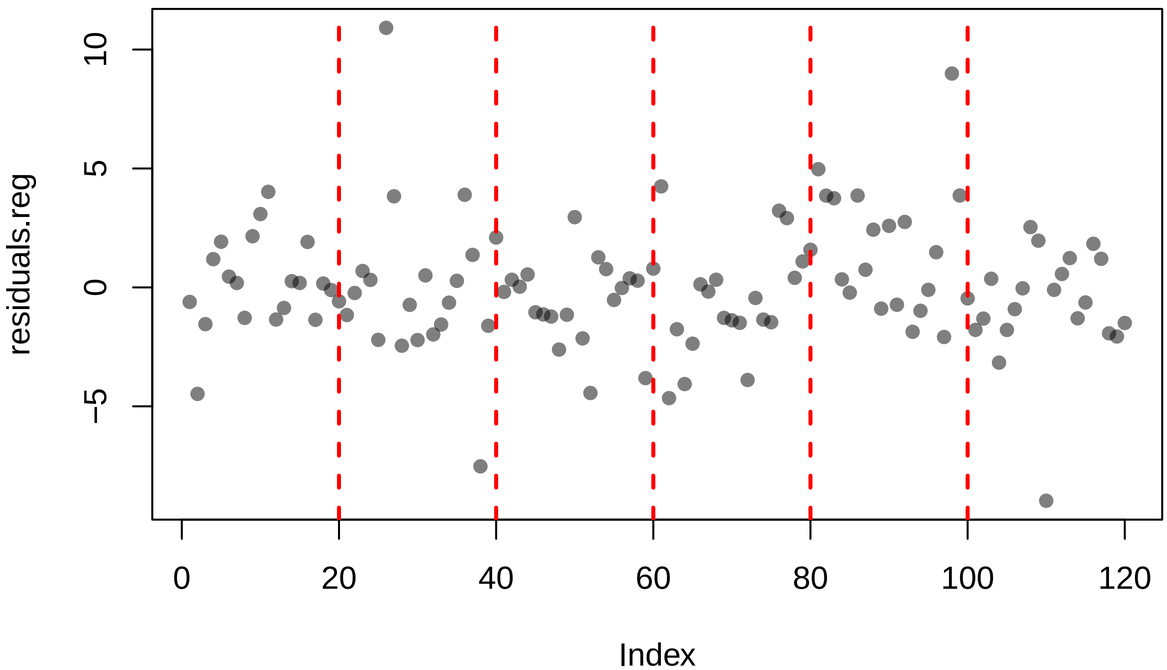
```
plot(residuals.reg, ylab = "Residuals for reg_2 model", main = "Residuals plot")
abline(h=0, col="red", lty=2, lwd=2)
legend(0,10,legend=c("zero-line"), col="red", lty=2, cex=.75)
```



From visual inspection, the residuals seem to have zero mean as they look evenly spread around the zero line. So this assumption **is satisfied**.

Constant variance:

```r
# Testing Constant variance with FLigner-Killeem
#First, 6 groups of 24 obs.
plot(residuals.reg, pch=16 , col=adjustcolor("black",0.5)) # plotting the residuals vs time
abline(v=c(1:5)*20 , col="red" , lwd=2, lty=2)
```




```r
segments = factor(rep(1:6,each=20)) #creating 6 chunks to test variance homogeneity
fligner.test(residuals.reg , segments) # Fligner's test for H0:sigma1=sigma2=...=sigma6 (note that we m
```

```
##
##  Fligner-Killeen test of homogeneity of variances
##
## data:  residuals.reg and segments
## Fligner-Killeen:med chi-squared = 6.7425, df = 5, p-value = 0.2405
```

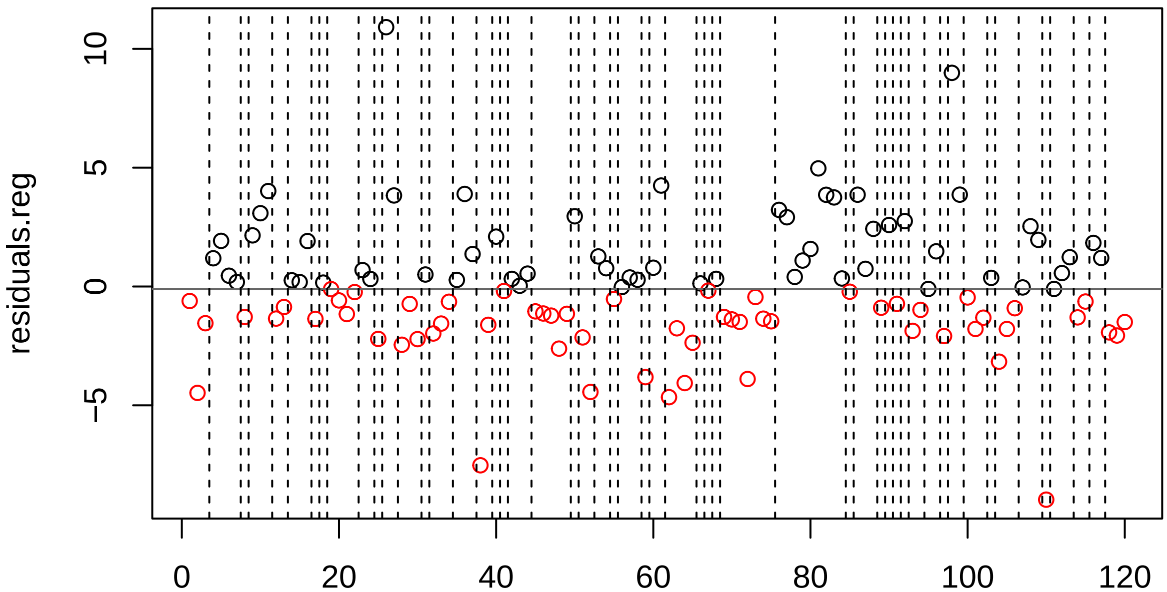
The $p-$value is high at 0.2405, so there is no evidence against the null hypothesis of equal variance. we conclude the constant variance assumption **is satisfied**.

Randomness (iid)

```r
# Testing randomness with difference sign test and runs test
randtests::difference.sign.test(residuals.reg)
```

```
##
##  Difference Sign Test
##
## data:  residuals.reg
## statistic = -0.15746, n = 120, p-value = 0.8749
## alternative hypothesis: nonrandomness
```

```r
randtests::runs.test(residuals.reg , plot = TRUE)
```




```
##
##  Runs Test
##
## data:  residuals.reg
## statistic = -1.4668, runs = 53, n1 = 60, n2 = 60, n = 120, p-value =
## 0.1424
## alternative hypothesis: nonrandomness
```

The *Difference Sign Test* has a large $p-$value at 0.8749 which means there was no evidence to reject the null hypothesis of random residuals. Therefore under this test, the randomness assumption **is satisfied.**

The *Runs test* has a $p-$value smaller than that of the DST but still large enough to suggest there is no evidence against the assumption of randomness. Therefore under this test, the randomness assumption **is satisfied.**

The normality assumption was not satisfied, so we should not make inferences with this model. Maybe trying make the mean stationary would solve this.