STA237-hw1

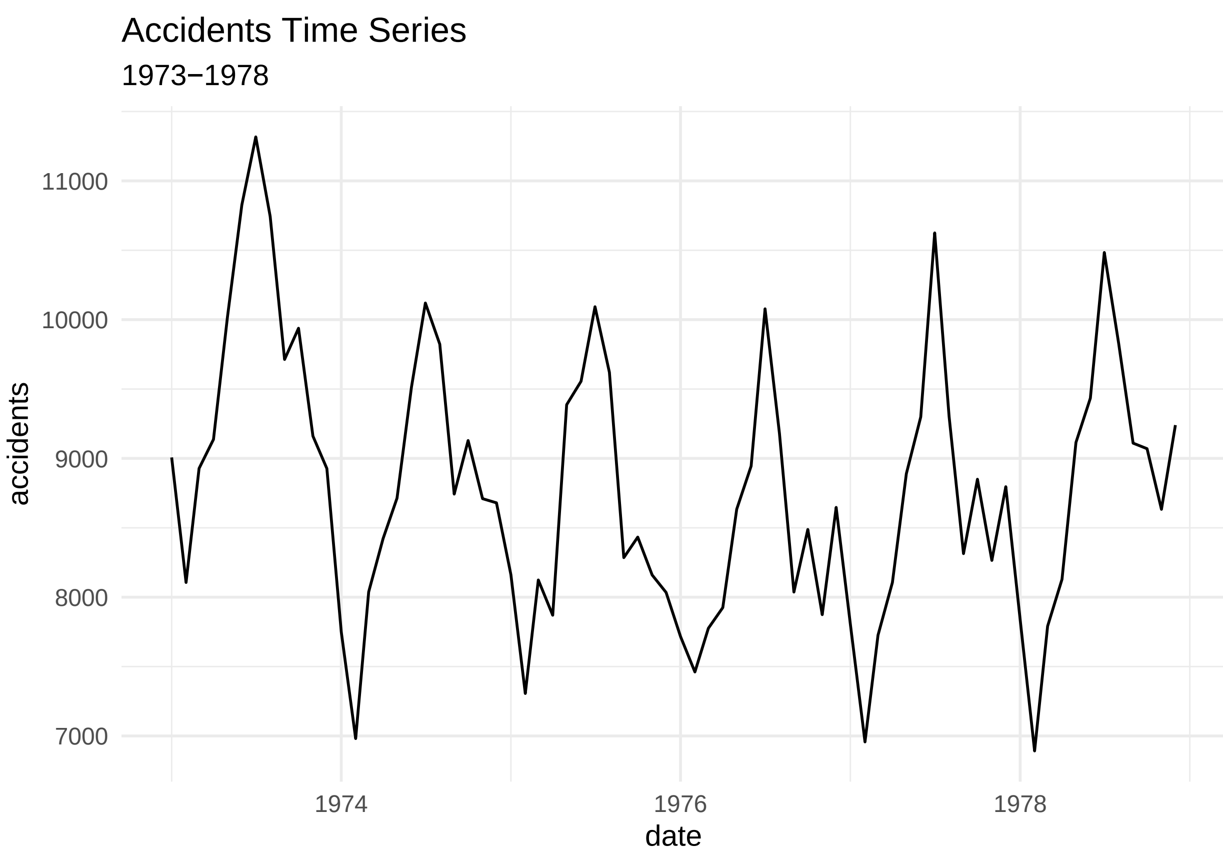
Collin Kennedy and Qianhui Wan

10/1/2021

Problem 2:

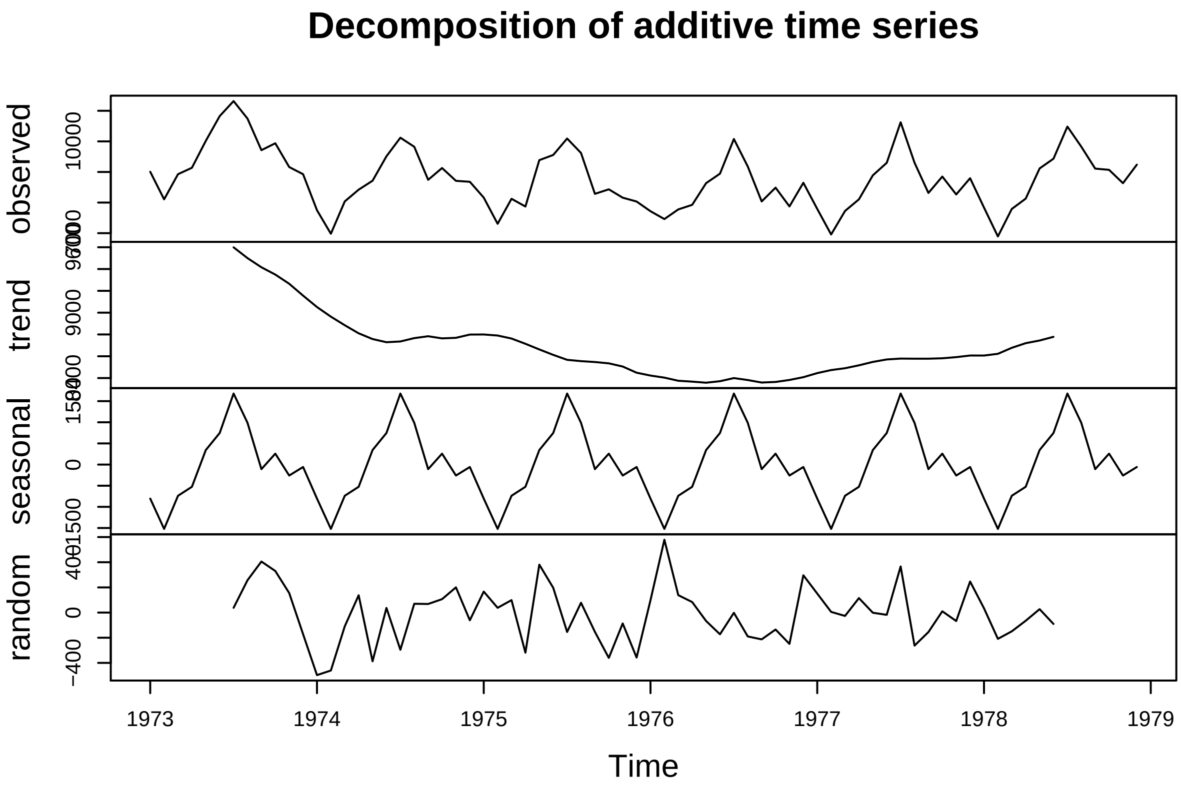
1) Plot the data;

```
accidents_ts = ts(data = accidents_data, start = c(1973,1), end = c(1978,12), frequency = 12)


accidents_data = accidents_data %>%
  mutate(date = seq.Date(from = my("011973"),to = my("121978"),by = "month"))


ggplot(data = accidents_data, mapping = aes(x = date,y=accidents))+
  geom_line()+
  ggtitle(label = "Accidents Time Series", subtitle = "1973-1978")+
  theme_minimal()
```



2) Find estimates st ,t= 1,. . . ,12, for the classical decompositionXt=mt+st+Zt, where

```
ts_components = decompose(accidents_ts)

plot(ts_components)
```




```
#we need to detrend before we can deseasonalize, correct? YES
#manually doing the decomposition:
accidents_data = accidents_data %>%
  mutate(year = format(date, "%Y")) %>%
  mutate(month = month(date))



accidents_transposed = accidents_data %>%
  select(accidents,year,month) %>%
  group_by(year) %>%
  summarise(mj_hat = mean(accidents))


full_accidents_df = accidents_data %>%
  select(-date) %>%
  pivot_wider(names_from = month,values_from = accidents) %>%
  left_join(accidents_transposed) %>%
  mutate(jan_diff = `1` - mj_hat) %>%
  mutate(feb_diff = `2` - mj_hat) %>%
  mutate(march_diff = `3` - mj_hat) %>%
  mutate(april_diff = `4` - mj_hat) %>%
```

```r
  mutate(may_diff = `5` - mj_hat) %>%
  mutate(june_diff = `6` - mj_hat) %>%
  mutate(july_diff = `7` - mj_hat) %>%
  mutate(aug_diff = `8` - mj_hat) %>%
  mutate(sept_diff = `9`- mj_hat) %>%
  mutate(oct_diff = `10` - mj_hat) %>%
  mutate(nov_diff = `11` - mj_hat) %>%
  mutate(dec_diff = `12` - mj_hat) %>%
  summarise(jan_comp = mean(jan_diff),feb_comp = mean(feb_diff),
            march_comp = mean(march_diff), april_comp = mean(april_diff),
            may_comp = mean(may_diff), june_comp = mean(june_diff),
            july_comp = mean(july_diff), aug_comp = mean(aug_diff),
            sept_comp = mean(sept_diff), oct_comp = mean(oct_diff),
            nov_comp = mean(nov_diff), dec_comp = mean(dec_diff))
```

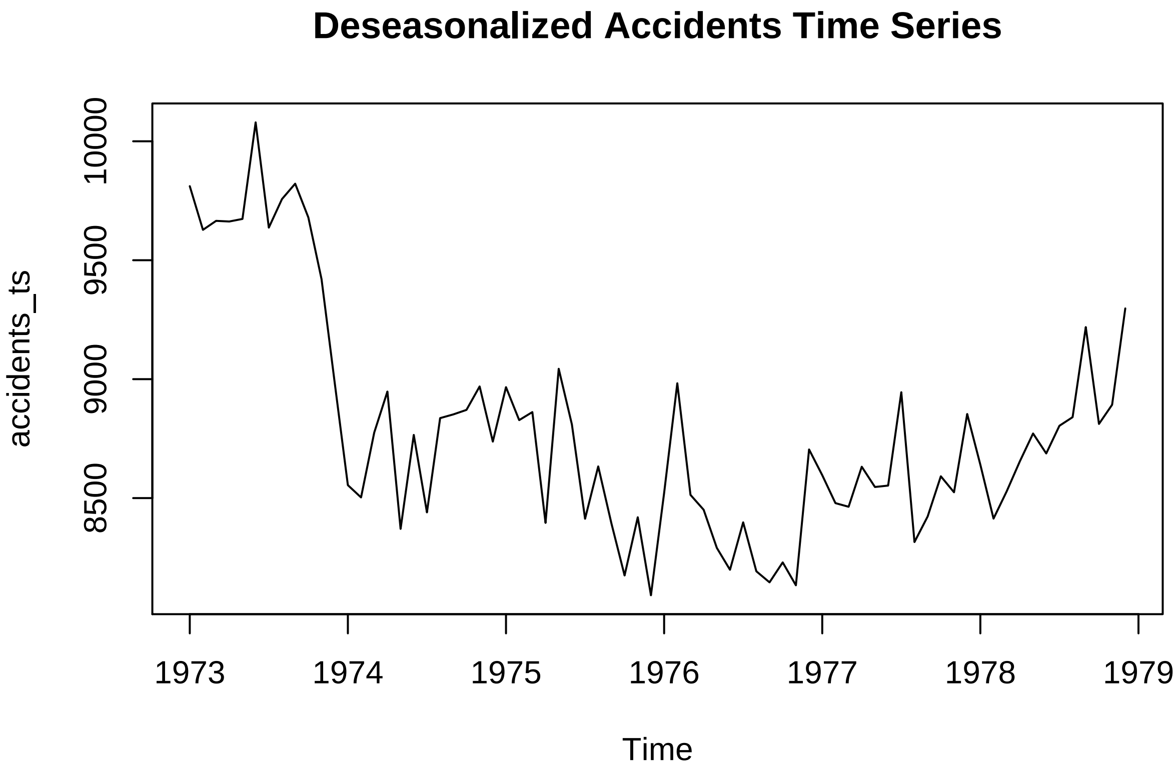
```
## Joining, by = "year"
```

```r
#confirmed manually




ts_components$seasonal
```

```
##             Jan         Feb         Mar         Apr         May         Jun
## 1973  -804.31944 -1521.73611  -737.46944  -525.81111   343.42222   746.41389
## 1974  -804.31944 -1521.73611  -737.46944  -525.81111   343.42222   746.41389
## 1975  -804.31944 -1521.73611  -737.46944  -525.81111   343.42222   746.41389
## 1976  -804.31944 -1521.73611  -737.46944  -525.81111   343.42222   746.41389
## 1977  -804.31944 -1521.73611  -737.46944  -525.81111   343.42222   746.41389
## 1978  -804.31944 -1521.73611  -737.46944  -525.81111   343.42222   746.41389
##             Jul         Aug         Sep         Oct         Nov         Dec
## 1973  1679.96389   986.83889  -108.76944   258.30556  -259.37778   -57.46111
## 1974  1679.96389   986.83889  -108.76944   258.30556  -259.37778   -57.46111
## 1975  1679.96389   986.83889  -108.76944   258.30556  -259.37778   -57.46111
## 1976  1679.96389   986.83889  -108.76944   258.30556  -259.37778   -57.46111
## 1977  1679.96389   986.83889  -108.76944   258.30556  -259.37778   -57.46111
## 1978  1679.96389   986.83889  -108.76944   258.30556  -259.37778   -57.46111
```

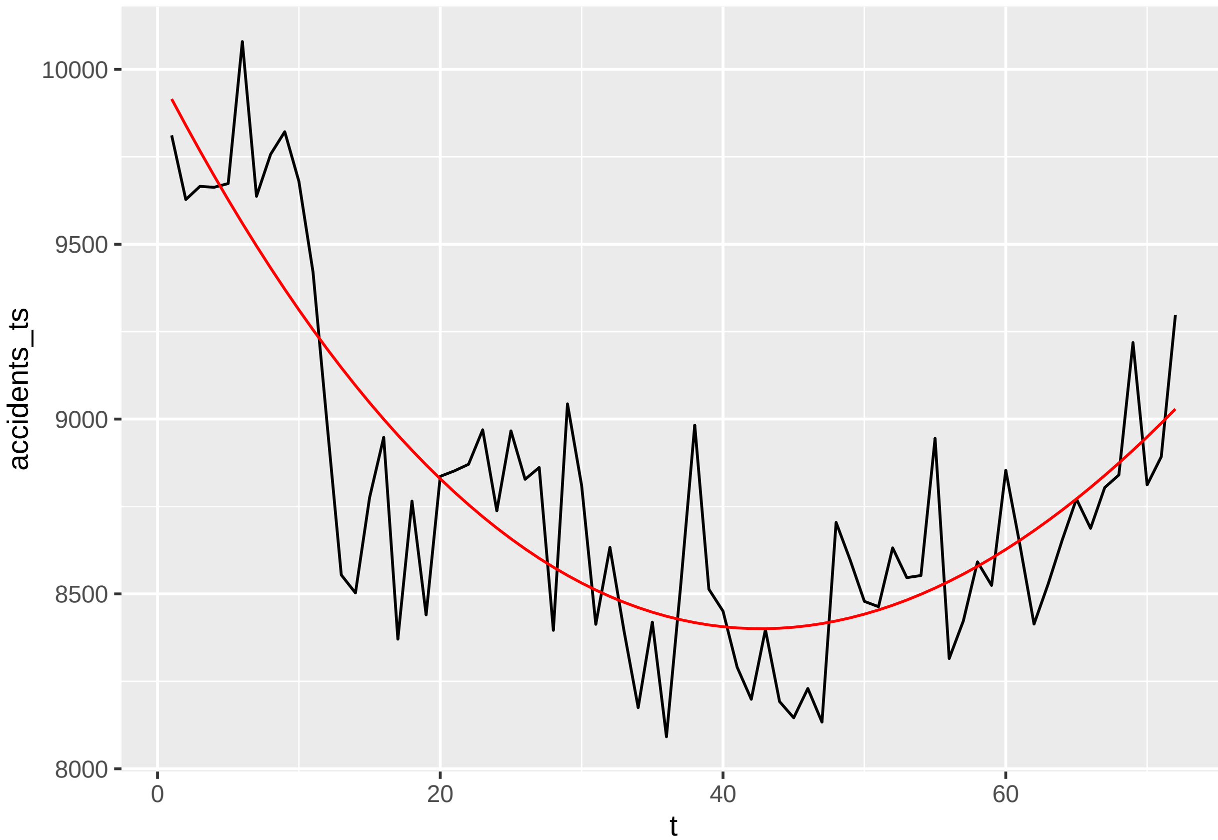
3) Plot the deseasonalized data

```r
deseasonalized_ts = accidents_ts - ts_components$seasonal

plot(deseasonalized_ts, main = "Deseasonalized Accidents Time Series")
```



4) Fit a suitable polynomial by least squares to the deseasonalized data and use it as your estimate $\hat{}$mt of mt;

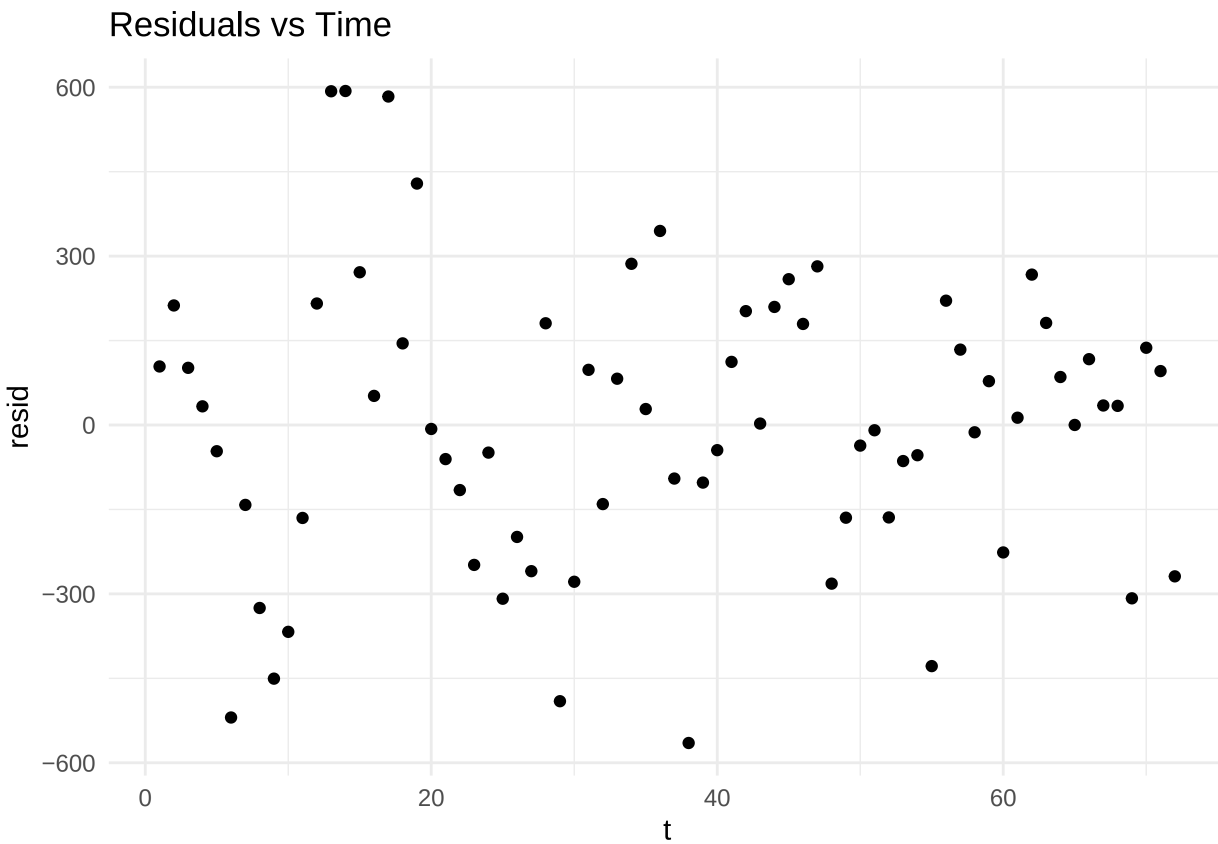
```
#get the fitted values from the polynomial model then plot that:
#fit a polynomial of order n = 3

poly_model = lm(accidents_ts ~ poly(t,3),data = deseasonalized_ts_df)
deseasonalized_ts_df = deseasonalized_ts_df %>%
  mutate(poly_fitted_values = poly_model$fitted.values)
ggplot(data = deseasonalized_ts_df, mapping =aes(x = t, y = accidents_ts))+
  geom_line()+
  geom_line(color = 'red', data = deseasonalized_ts_df, mapping = aes(x = t, y = poly_fitted_values ))
```



Here I plot a polynomial of order 3. It appears to fit the overall trend of the data without overfitting.

5) Plot the residuals

```
#note that accidents_ts is already adjusted for the seasonal component
deseasonalized_ts_df = deseasonalized_ts_df %>%
  mutate(resid = poly_fitted_values - accidents_ts) %>%
  mutate(std_residuals = resid/sd(resid))

ggplot(data = deseasonalized_ts_df, mapping = aes(x = t, y = resid))+
  geom_point()+
  ggtitle(label = "Residuals vs Time")+
  theme_minimal()
```



The residuals appear to be somewhat randomly distributed about 0, indicating constant variance.

6) Compute the sample ACF of the residuals

```
acf(deseasonalized_ts_df$resid,plot = F)
```

```
##
## Autocorrelations of series 'deseasonalized_ts_df$resid', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000  0.382  0.230  0.144 -0.091 -0.047 -0.175 -0.340 -0.337 -0.211 -0.215
##     11     12     13     14     15     16     17     18
## -0.099 -0.084 -0.143  0.044  0.007  0.020  0.189 -0.004
```

7) Use your fitted model to predict Xt

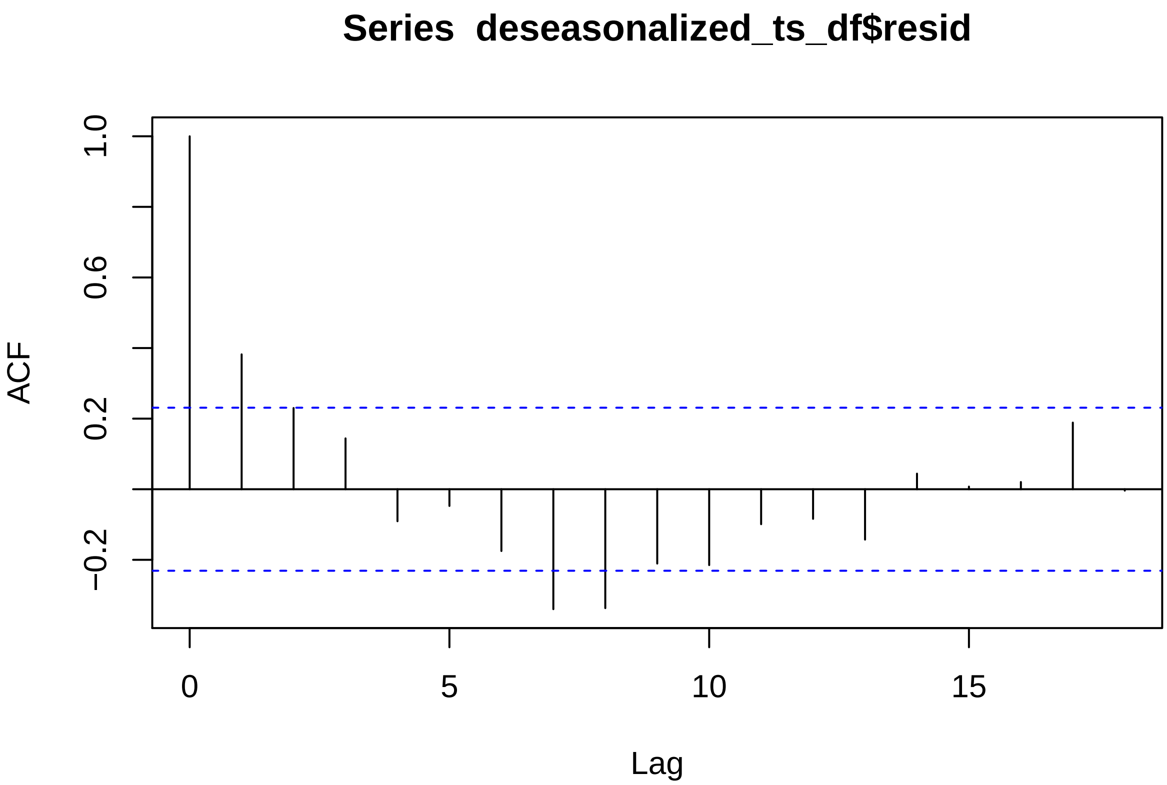
```
#Right now I've essentially regressed  seasonally adjusted accidents on an nth order polynomial. is tha


new_data = seq(from = 73, to = 84, by = 1)
new_data = tibble(t = new_data)
model_predictions = predict(poly_model,newdata = new_data)
```

Our third order polynomial model predicts the following number of accidents for t = 73,. . . , 84: 9070.3521895, 9113.2717417, 9157.3914101, 9202.6991432, 9249.1828897, 9296.830598, 9345.6302169, 9395.5696948, 9446.6369803, 9498.8200221, 9552.1067687, 9606.4851687

Problem 3: Testing the Residuals

Method 1: Sample ACF

```r
acf(deseasonalized_ts_df$resid)
```



The ACF plot above is a good tool to use to get an initial idea of the autocorrelation (or lack thereof) that exists in the time series. For the most part this sample ACF plot seems pretty good, in the sense that most of the lags have statistically *in*significant autocorrelation. However, there are a couple lags (between 5 and 10) that appear to be significant. This is good justification for further exploration of the autocorrelation in the data.

Method 2: Portmanteau Test

H_0: independent and identically distributed residuals

H_a: the residuals are not independent and identically distributed

```r
Box.test(deseasonalized_ts_df$resid, type = "Ljung-Box", fitdf = 0)
```

```
##
##  Box-Ljung test
##
## data:  deseasonalized_ts_df$resid
## X-squared = 10.961, df = 1, p-value = 0.0009307
```

```r
acf_values = acf(deseasonalized_ts_df$resid,plot = F)$acf

acf(deseasonalized_ts_df$resid,plot = F)
```

```
##
## Autocorrelations of series 'deseasonalized_ts_df$resid', by lag
```

```
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000  0.382  0.230  0.144 -0.091 -0.047 -0.175 -0.340 -0.337 -0.211 -0.215
##     11     12     13     14     15     16     17     18
## -0.099 -0.084 -0.143  0.044  0.007  0.020  0.189 -0.004
```

```r
sum_rho_squared = 0
for(i in 2:18){
  rho_i_squared = (acf_values[i])^2
  sum_rho_squared = sum_rho_squared + rho_i_squared

}
Q_statistic = 72*sum_rho_squared


Q_statistic > qchisq(.95,18)
```

```
## [1] TRUE
```

```r
p_value = pchisq(Q_statistic,18,lower.tail = FALSE)
```

We calculate a p-value of $\sim 1.965549 \times 10^{-4}$, which is $< .05$, so we reject the null hypothesis. We are a little concerned with this result given what we find with Methods 3 and 4 (where we fail to reject the null of iid residuals). The results seem contradictory but we are unsure as to why.

Method 3: The Rank Test

H_0: residuals are independent and identically distributed

H_a: residuals are *not* independent and identically distributed

```r
rank_test_df = deseasonalized_ts_df %>%
  select(std_residuals) %>%
  mutate(i = seq(1,72,by = 1)) %>%
  mutate(j = seq(1,72,by = 1))

rank_test_df
```

```
## # A tibble: 72 x 3
##    std_residuals     i     j
##            <dbl> <dbl> <dbl>
##  1         0.415     1     1
##  2         0.847     2     2
##  3         0.405     3     3
##  4         0.132     4     4
##  5        -0.186     5     5
##  6        -2.07      6     6
##  7        -0.566     7     7
##  8        -1.30      8     8
##  9        -1.80      9     9
## 10        -1.47     10    10
## # ... with 62 more rows
```

```r
random_pairs_df = tibble(std_residual_i = numeric(),std_residual_j = numeric(),i = numeric(),j = numeri


for(i in 2:length(rank_test_df$std_residuals)){#start from the second row
```

```r
  for(j in 1:length(rank_test_df$std_residuals)){

    if(rank_test_df[i,]$i > rank_test_df[j,]$j){
      #add that pair to the random_pairs_df
      random_pairs_df = random_pairs_df %>% add_row(std_residual_i = rank_test_df$std_residuals[[i]],
                                    std_residual_j = rank_test_df$std_residuals[[j]],
                                    i = rank_test_df$i[[i]],
                                    j = rank_test_df$j[[j]]
                                    )
    }else{
      break
    }

  }
}

#calculate Pi
random_pairs_df %>%
  mutate(res_i_greater_res_j = ifelse(std_residual_i > std_residual_j,1,0)) %>%
  summarise(pi = sum(res_i_greater_res_j)) #pi = 1307
```

```
## # A tibble: 1 x 1
##      pi
##   <dbl>
## 1  1307
```

```r
mu_pi = (1/4)*length(rank_test_df$std_residuals)*(length(rank_test_df$std_residuals)-1)
sigma_pi = (1/72)*length(rank_test_df$std_residuals)*(length(rank_test_df$std_residuals)-1)*(2*length(ra
sigma_pi
```

```
## [1] 10579
```

```r
mu_pi
```

```
## [1] 1278
```

```r
P_statistic = (1307 - mu_pi)/sigma_pi
P_statistic
```

```
## [1] 0.00274128
```

```r
P_statistic > 1.96
```

```
## [1] FALSE
```

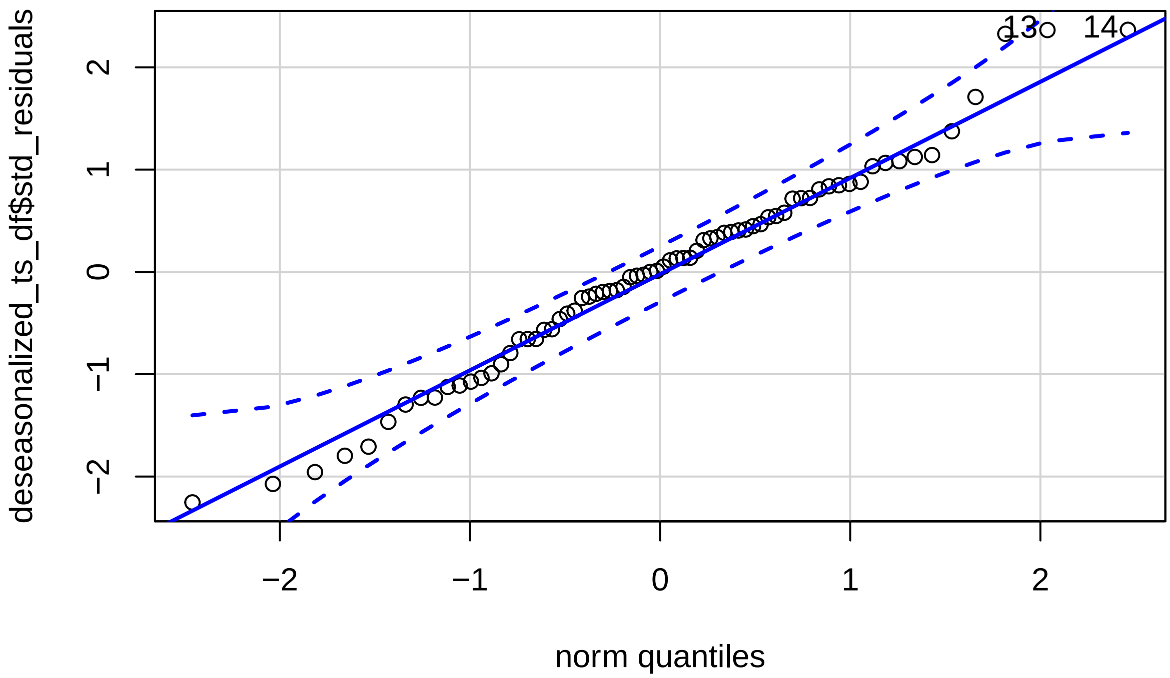
Since our test statistic P = 0.0027413 *is not* > 1.96, the corresponding normal distribution critical value at α = .05, we fail to reject the null hypothesis and conclude that the residuals are independent and identically distributed.

Method 4: QQPlot (Test for Normality)

```r
#standardize the residuals first
deseasonalized_ts_df = deseasonalized_ts_df %>%
  mutate(std_residuals = resid/sd(resid))

deseasonalized_ts_df
```

```
## # A tibble: 72 x 5
##    accidents_ts     t poly_fitted_values  resid std_residuals
##           <dbl> <dbl>              <dbl>  <dbl>         <dbl>
##  1         9811.     1              9915.  104.          0.415
##  2         9628.     2              9840.  212.          0.847
##  3         9665.     3              9767.  102.          0.405
##  4         9663.     4              9696.   33.2         0.132
##  5         9674.     5              9627.  -46.6        -0.186
##  6        10080.     6              9560. -520.         -2.07
##  7         9637.     7              9495. -142.         -0.566
##  8         9757.     8              9432. -325.         -1.30
##  9         9822.     9              9371. -451.         -1.80
## 10         9680.    10              9312. -367.         -1.47
## # ... with 62 more rows
```

```r
qqPlot(deseasonalized_ts_df$std_residuals)
```




```
## [1] 14 13
```

Since the vast majority of the residuals fall along the line of the qq plot, this tells us that it is appropriate to assume the residuals are approximately normally distributed. #explain qqplot