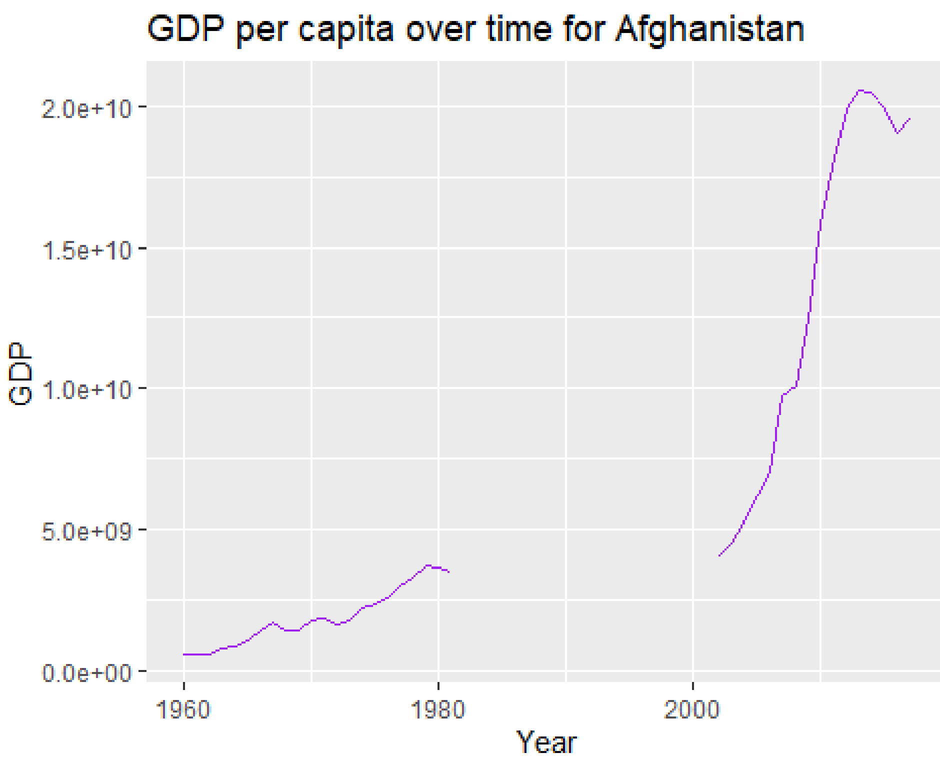
## Part 1

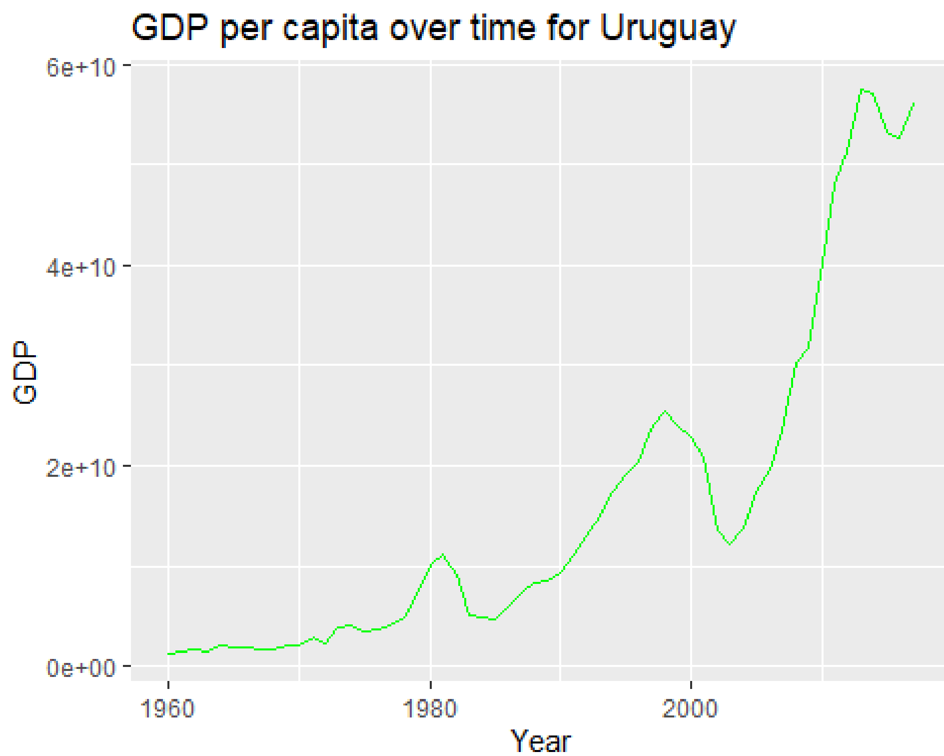
Loading the required package into Rstudio

```
# First 6 rows of our data set

head(global_economy)

## # A tsibble: 6 x 9 [1Y]
## # Key:       Country [1]
##   Country     Code   Year         GDP Growth   CPI Imports Exports Population
##   <fct>       <fct> <dbl>       <dbl>  <dbl> <dbl>   <dbl>   <dbl>      <dbl>
## 1 Afghanistan AFG    1960  537777811.     NA    NA    7.02    4.13    8996351
## 2 Afghanistan AFG    1961  548888896.     NA    NA    8.10    4.45    9166764
## 3 Afghanistan AFG    1962  546666678.     NA    NA    9.35    4.88    9345868
## 4 Afghanistan AFG    1963  751111191.     NA    NA   16.9     9.17    9533954
## 5 Afghanistan AFG    1964  800000044.     NA    NA   18.1     8.89    9731361
## 6 Afghanistan AFG    1965 1006666638.     NA    NA   21.4    11.3     9938414

# Selecting Vietnam, Uruguay, and Afghanistan countries.

# Afghanistan

Afghanistan_data <-global_economy %>% filter(Country=="Afghanistan")
```

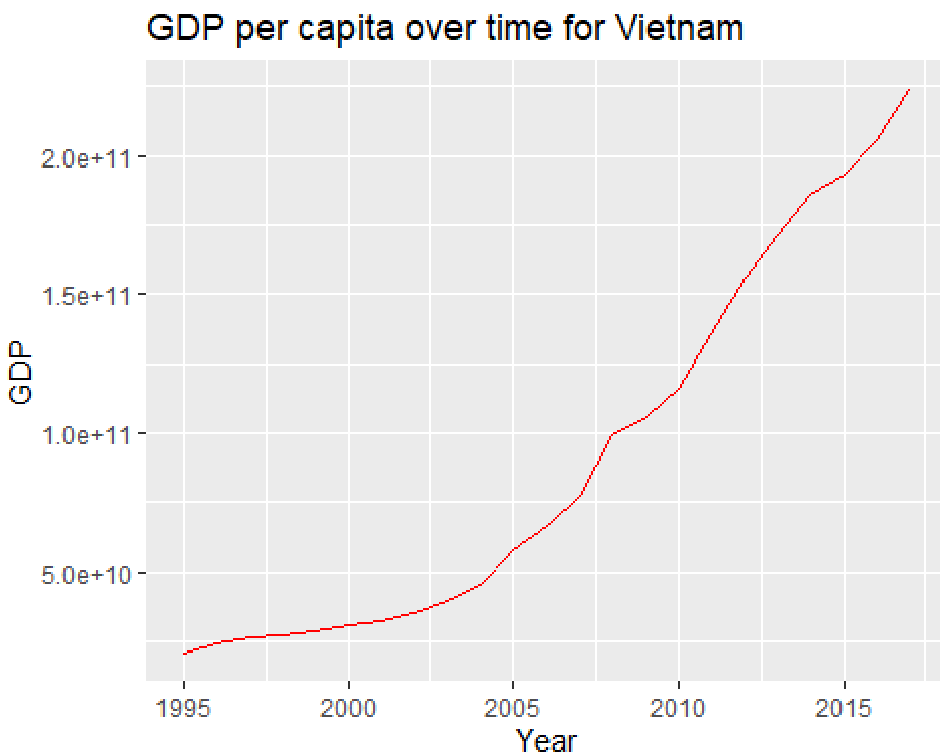
```
ggplot(data = Afghanistan_data,aes(x=Year,y=GDP))+geom_line(col="purple")+ggtitle("GDP per capita over time for Afghanistan")
```




```
#Uruguay
Uruguay_data<- global_economy %>% filter(Country=="Uruguay")
ggplot(data =Uruguay_data,aes(x=Year,y=GDP))+geom_line(col="green")+ggtitle("GDP per capita over time for Uruguay")
```




```
#Uruguay

# Vietnam

Vietnam_data<-global_economy %>% filter(Country=="Vietnam")

ggplot(data = na.omit(Vietnam_data),aes(x=Year,y=GDP))+geom_line(col="red")+ggtitle("GDP per capita over time for Vietnam")
```




Which has highest GDP per capita?

```
options(scipen = 999)

sum(Afghanistan_data$GDP,na.rm=T)

## [1] 253490832378

sum(Uruguay_data$GDP,na.rm=T)

## [1] 931336081676

sum(Vietnam_data$GDP,na.rm=T)

## [1] 2269838130133

# Based on the above results, Vietnam has the highest GDP.
```

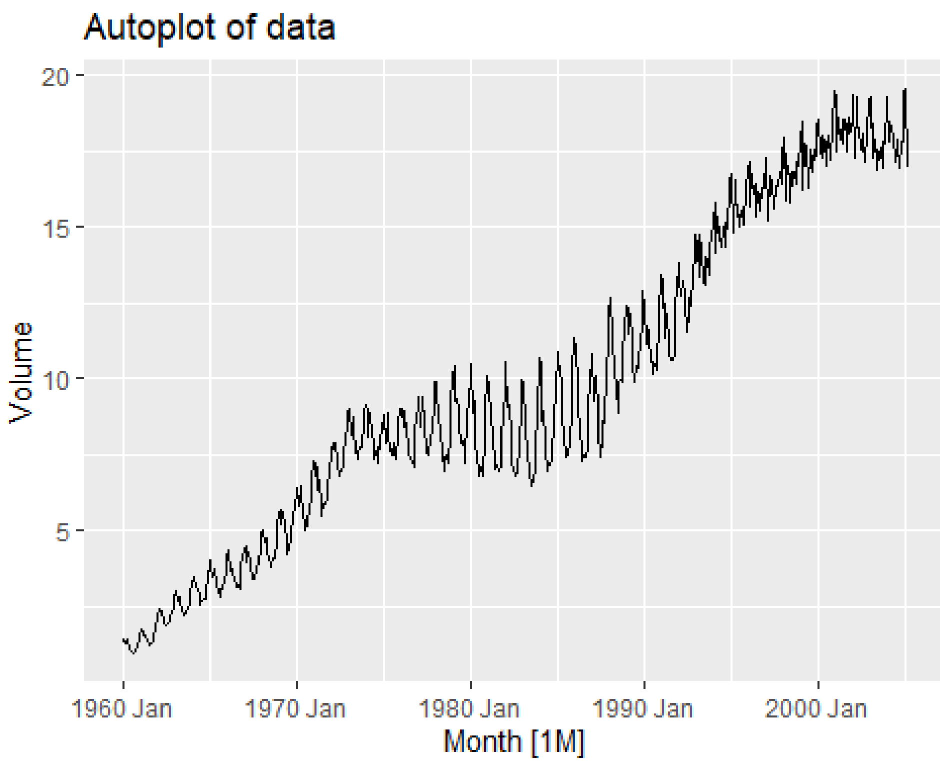
Checking on the above plots, it is visible that GDP has increased over the years.

## PART 2.

Use the canadian_gas data (monthly Canadian gas production in billions of cubic metres, January 1960 – February 2005).

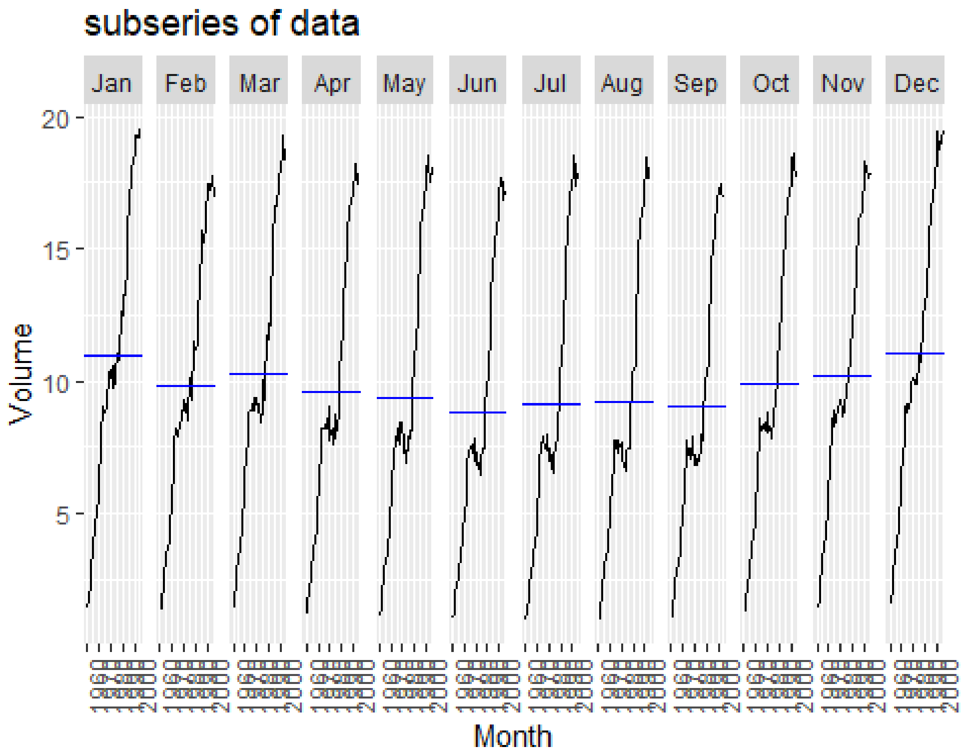
```
str(canadian_gas)

## tbl_ts [542 x 2] (S3: tbl_ts/tbl_df/tbl/data.frame)
##  $ Month : mth [1:542] 1960 Jan, 1960 Feb, 1960 Mar, 1960 Apr, 1960 May, 1960 Jun...
##  $ Volume: num [1:542] 1.43 1.31 1.4 1.17 1.12 ...
##  - attr(*, "key")= tibble [1 x 1] (S3: tbl_df/tbl/data.frame)
##   ..$ .rows: list<int> [1:1]
##   .. ..$ : int [1:542] 1 2 3 4 5 6 7 8 9 10 ...
##   .. ..@ ptype: int(0)
##  - attr(*, "index")= chr "Month"
##   ..- attr(*, "ordered")= logi TRUE
##  - attr(*, "index2")= chr "Month"
##  - attr(*, "interval")= interval [1:1] 1M
##   ..@ .regular: logi TRUE

# Canadian_gas data contains 542 observations of 2 variables.
# Plot Volume using autoplot, gg_subseries, gg_season to look at the effect of changing seasonality over time
autoplot(canadian_gas)+labs(title = "Autoplot of data")
```

```
## Plot variable not specified, automatically selected `.vars = Volume`
```

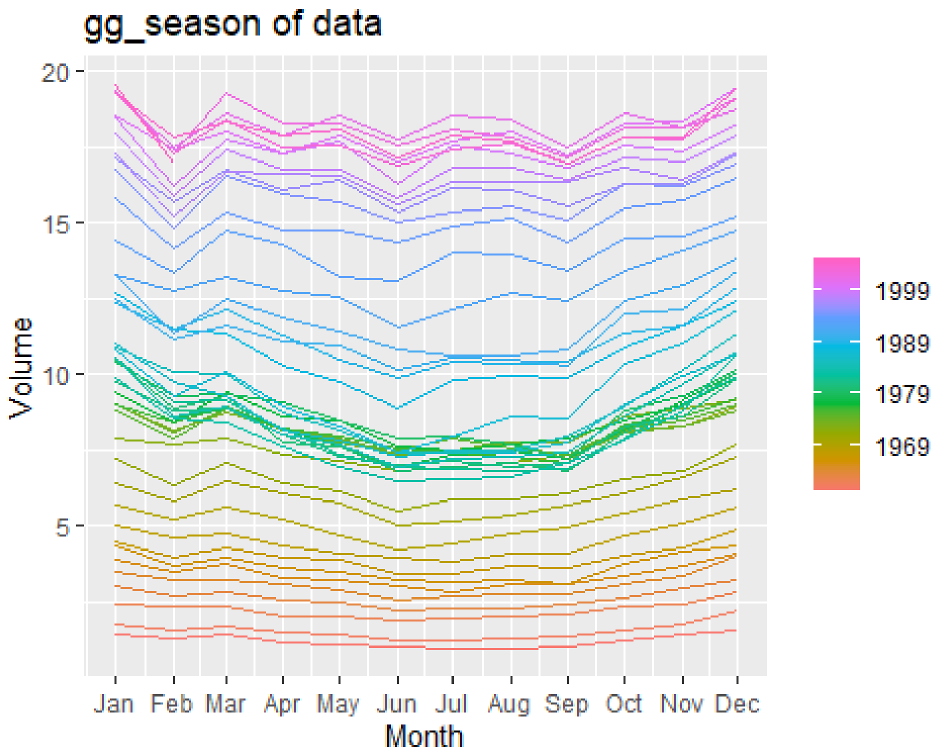



```
gg_subseries(canadian_gas)+ labs(title = "subseries of data")
```

```
## Plot variable not specified, automatically selected `y = Volume`
```




```
gg_season(canadian_gas) + labs(title = "gg_season of data")

## Plot variable not specified, automatically selected `y = Volume`
```

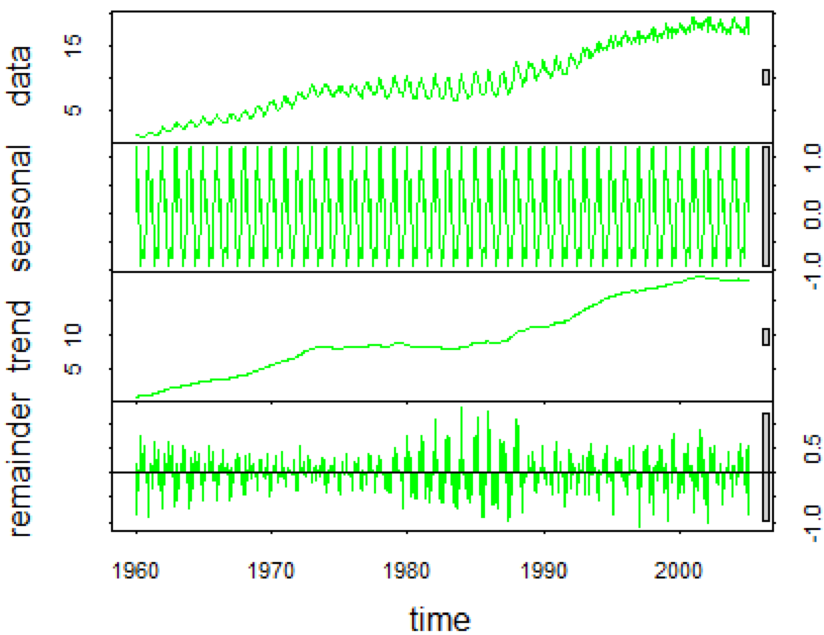



Do an STL decomposition of the data. You will need to choose a seasonal window to allow for the changing shape of the seasonal component.

```
decomposition<-stl(canadian_gas,s.window = "periodic")
```

How does seasonal shape change over time? [Hint: Try plotting the seasonal component using gg_season().]

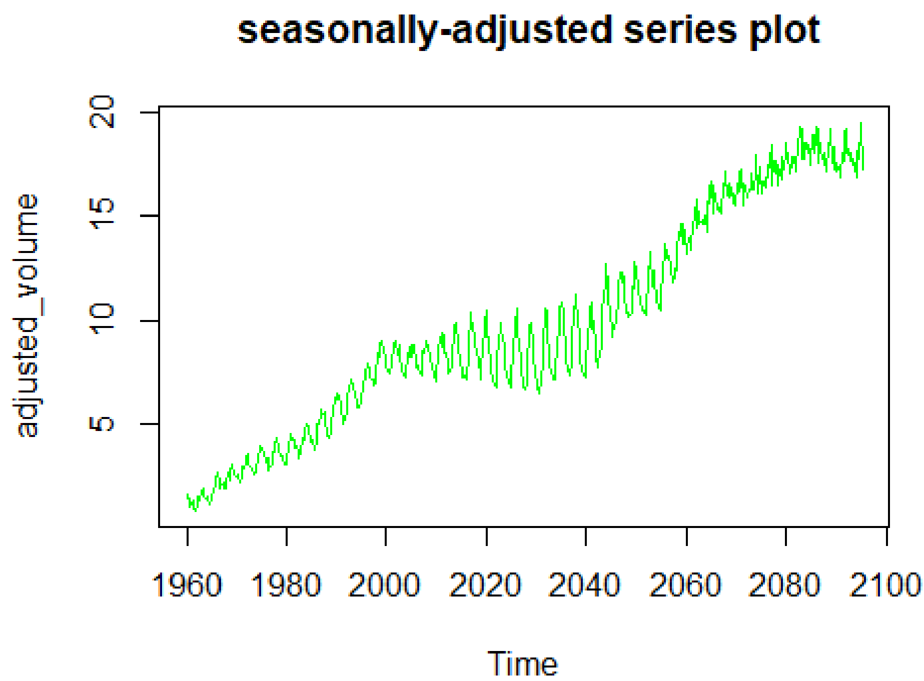
```
plot(decomposition,col="green")
```




```
# Under the seasonal plot above, we can say that the data exhibits a seasonal pattern
```

Can you produce a plausible seasonally adjusted series?

```
sas<-as.ts(canadian_gas,frequency = 4,start=1990,end=2025)

dec_adjust<-decompose(sas,type="additive")

adjusted_volume<-sas-dec_adjust$seasonal

plot(adjusted_volume,col="green",main=" seasonally-adjusted series plot")
```




```
#plausible seasonally adjusted series analyses trends by removing all the noises present.
```

**PART 3**

Aus Retail Time Series We will use aus_rail dataset Using the code below, get a series (it gets a series randomly by using sample() function):

```
set.seed(1234567)
myseries <- aus_retail %>%
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))
head(myseries)

## # A tsibble: 6 x 5 [1M]
## # Key:       State, Industry [1]
##   State    Industry                                   Serie~1    Month Turno~2
```

```
##   <chr>    <chr>                                 <chr>      <mth>   <dbl>
## 1 Victoria Cafes, restaurants and takeaway food servic~ A33494~ 1982 Apr    85.1
## 2 Victoria Cafes, restaurants and takeaway food servic~ A33494~ 1982 May    85.1
## 3 Victoria Cafes, restaurants and takeaway food servic~ A33494~ 1982 Jun    82.8
## 4 Victoria Cafes, restaurants and takeaway food servic~ A33494~ 1982 Jul    82.1
## 5 Victoria Cafes, restaurants and takeaway food servic~ A33494~ 1982 Aug    81.8
## 6 Victoria Cafes, restaurants and takeaway food servic~ A33494~ 1982 Sep    84.6
## # ... with abbreviated variable names 1: `Series ID`, 2: Turnover
```

```
# remover NA's in the series with below:
myseries =  myseries %>% filter(!is.na(`Series ID`))
nrow(myseries)
```

```
## [1] 441
```

```
# rename the column name  `Series ID` with MyRandomSeries
rename(myseries, MyRandomSeries = `Series ID`)
```

```
## # A tsibble: 441 x 5 [1M]
## # Key:       State, Industry [1]
##    State    Industry                                MyRan~1    Month Turno~2
##    <chr>    <chr>                                   <chr>      <mth>   <dbl>
##  1 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Apr    85.1
##  2 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 May    85.1
##  3 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Jun    82.8
##  4 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Jul    82.1
```

```
##  5 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Aug    81.8
##  6 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Sep    84.6
##  7 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Oct    91.7
##  8 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Nov    97.7
##  9 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1982 Dec   109.
## 10 Victoria Cafes, restaurants and takeaway food servi~ A33494~ 1983 Jan    94.6
## # ... with 431 more rows, and abbreviated variable names 1: MyRandomSeries,
## #   2: Turnover
```

a) Run a linear regression of Turnover on its trend. Hint: use TSLM() and trend() functions)

```
fit = myseries %>% model(TSLM(Turnover~ trend()))
report(fit)

## Series: Turnover
## Model: TSLM
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -125.471  -50.951   -9.889   48.598  242.364
##
## Coefficients:
##             Estimate Std. Error t value             Pr(>|t|)
## (Intercept)  -23.529      6.121  -3.844             0.000139 ***
## trend()        1.921      0.024  80.057 < 0.0000000000000002 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 64.17 on 439 degrees of freedom
## Multiple R-squared: 0.9359,  Adjusted R-squared: 0.9357
## F-statistic:  6409 on 1 and 439 DF, p-value: < 0.0000000000000000222

# checking o the value of Multiple R-squared, it means that our model is 93.59 accurate
```

b) Forecast for next 3 years. What are the values for the next 3 years? Monthly values?

```
x<-myseries$Turnover %>% forecast(h=36)
x

##     Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 442       1014.052 926.9777 1101.126 880.8834 1147.221
## 443       1015.375 918.7325 1112.017 867.5733 1163.176
## 444       1016.697 911.3192 1122.076 855.5354 1177.859
## 445       1018.020 904.5455 1131.495 844.4756 1191.565
## 446       1019.343 898.2828 1140.403 834.1974 1204.488
## 447       1020.666 892.4403 1148.891 824.5620 1216.769
## 448       1021.988 886.9512 1157.025 815.4669 1228.510
## 449       1023.311 881.7643 1164.858 806.8341 1239.788
## 450       1024.634 876.8397 1172.428 798.6022 1250.665
## 451       1025.956 872.1452 1179.768 790.7225 1261.190
## 452       1027.279 867.6549 1186.903 783.1550 1271.403
## 453       1028.602 863.3472 1193.857 775.8667 1281.337
```

```
## 454      1029.925 859.2039 1200.645 768.8298 1291.019
## 455      1031.247 855.2096 1207.285 762.0209 1300.474
## 456      1032.570 851.3513 1213.789 755.4199 1309.720
## 457      1033.893 847.6175 1220.168 749.0093 1318.776
## 458      1035.215 843.9982 1226.433 742.7739 1327.657
## 459      1036.538 840.4849 1232.591 736.7006 1336.376
## 460      1037.861 837.0698 1238.652 730.7774 1344.944
## 461      1039.184 833.7460 1244.621 724.9939 1353.373
## 462      1040.506 830.5075 1250.505 719.3408 1361.672
## 463      1041.829 827.3488 1256.309 713.8098 1369.848
## 464      1043.152 824.2649 1262.039 708.3932 1377.910
## 465      1044.474 821.2514 1267.698 703.0842 1385.865
## 466      1045.797 818.3042 1273.290 697.8767 1393.718
## 467      1047.120 815.4196 1278.820 692.7649 1401.475
## 468      1048.443 812.5943 1284.291 687.7437 1409.142
## 469      1049.765 809.8251 1289.706 682.8084 1416.722
## 470      1051.088 807.1091 1295.067 677.9545 1424.222
## 471      1052.411 804.4438 1300.378 673.1780 1431.644
## 472      1053.734 801.8267 1305.640 668.4752 1438.992
## 473      1055.056 799.2555 1310.857 663.8427 1446.270
## 474      1056.379 796.7281 1316.030 659.2773 1453.481
## 475      1057.702 794.2426 1321.161 654.7758 1460.628
```

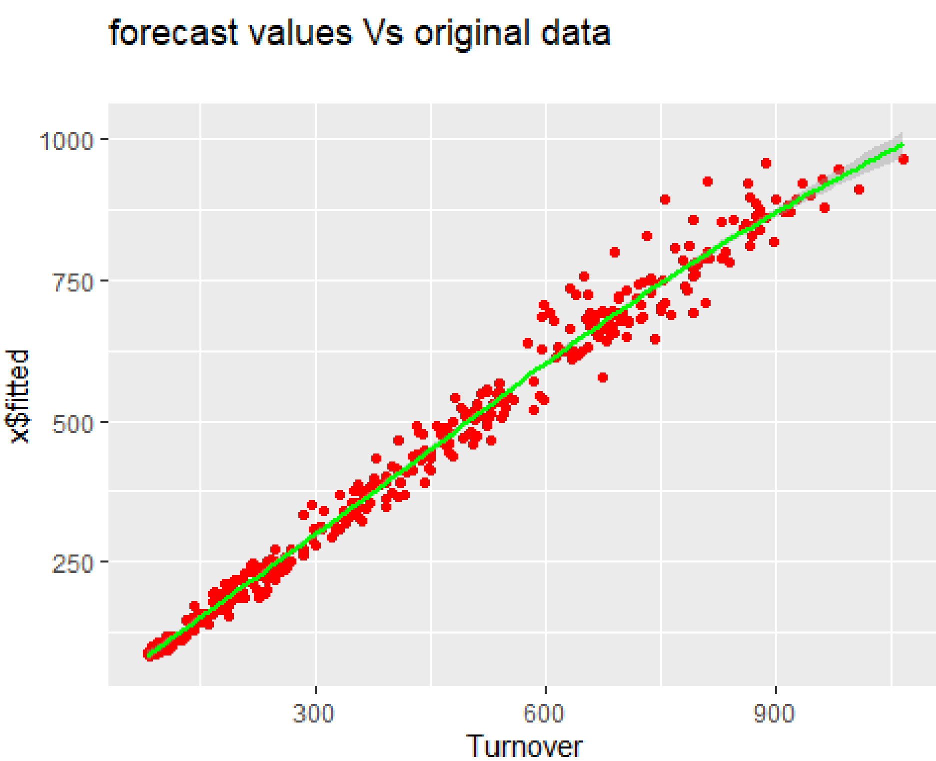
## 476       1059.024 791.7972 1326.252 650.3357 1467.713

## 477       1060.347 789.3901 1331.304 645.9542 1474.740

# In the table above Print Forecast column shows the forecasted values of the next 3 years

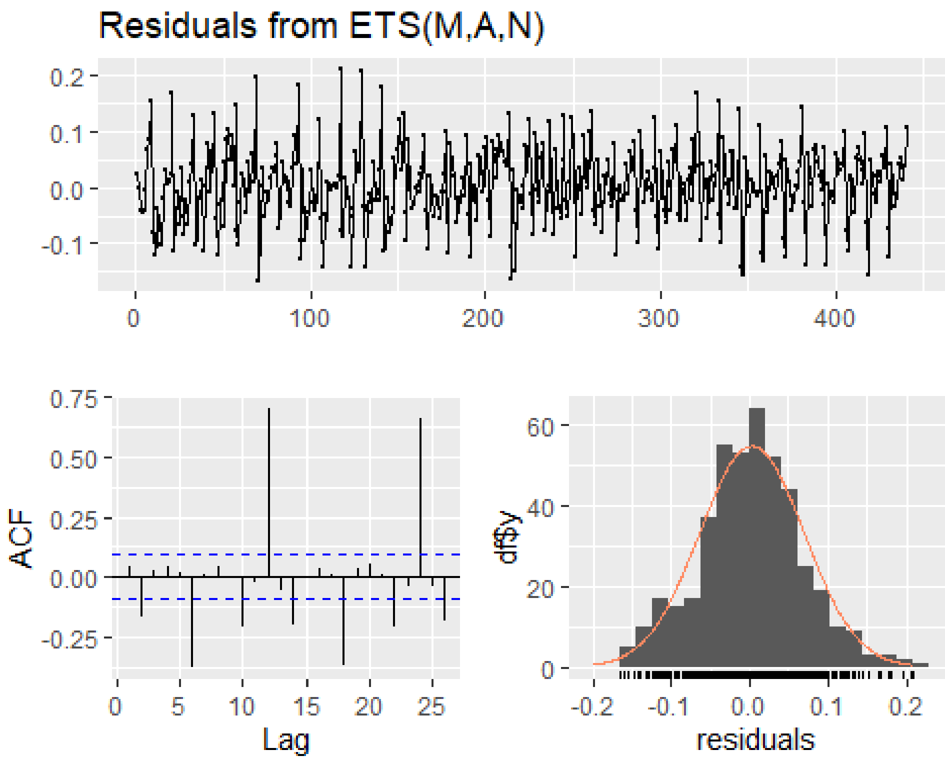
c) Plot the forecast values with original data

```
ggplot(data =
myseries,aes(Turnover,x$fitted))+geom_point(col="red")+geom_smooth(col="green")+ggtitle("f
orecast values Vs original data
")

## Don't know how to automatically pick scale for object of type <ts>. Defaulting
## to continuous.
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```



d) Get the residuals, does it satisfy requirements for white noise error terms. Hint: augment() and gg_tsresiduals() functions)

```
checkresiduals(x,plot = T)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,N)
## Q* = 97.597, df = 6, p-value < 0.00000000000000022
##
## Model df: 4.   Total lags used: 10
```

# By checking the p-value and the plot above, the residuals satisfy the requirements for white noise error terms.