Tratamento inicial dos dados

```
data = read.csv('Month_Value_1.csv')
head(data)
```

```
##       Period  Revenue Sales_quantity Average_cost
## 1 01.01.2015 16010072          12729      1257.764
## 2 01.02.2015 15807587          11636      1358.567
## 3 01.03.2015 22067146          15922      1384.697
## 4 01.04.2015 18016583          13227      1361.667
## 5 01.05.2015 14614488           9518      1535.422
## 6 01.06.2015 16783139          12227      1371.175
##   The_average_annual_payroll_of_the_region
## 1                                 30024676
## 2                                 30024676
## 3                                 30024676
## 4                                 30024676
## 5                                 30024676
## 6                                 30024676
```

Period está mal formatado, devemos transformar para formato de data.

Vamos renomear as variáveis.

```
names(data) = c('Period', 'Revenue', 'Sales', 'Average_cost', 'Average_payroll')
dataPeriod = as.Date(data$Period, format = "%d.%m.%Y")
is.na(data) %>% summary()
```

```
##    Period          Revenue          Sales        Average_cost
##  Mode :logical   Mode :logical   Mode :logical   Mode :logical
##  FALSE:96        FALSE:64        FALSE:64        FALSE:64
##                  TRUE :32        TRUE :32        TRUE :32
##  Average_payroll
##  Mode :logical
##  FALSE:64
##  TRUE :32
```

```
data[!complete.cases(data),] # e partir da linha 65 todos as linhas são NA. Vamos removê-las
```

```
##        Period Revenue Sales Average_cost Average_payroll
## 65 2020-05-01      NA    NA           NA              NA
## 66 2020-06-01      NA    NA           NA              NA
## 67 2020-07-01      NA    NA           NA              NA
## 68 2020-08-01      NA    NA           NA              NA
## 69 2020-09-01      NA    NA           NA              NA
## 70 2020-10-01      NA    NA           NA              NA
## 71 2020-11-01      NA    NA           NA              NA
## 72 2020-12-01      NA    NA           NA              NA
## 73 2021-01-01      NA    NA           NA              NA
## 74 2021-02-01      NA    NA           NA              NA
## 75 2021-03-01      NA    NA           NA              NA
## 76 2021-04-01      NA    NA           NA              NA
## 77 2021-05-01      NA    NA           NA              NA
## 78 2021-06-01      NA    NA           NA              NA
## 79 2021-07-01      NA    NA           NA              NA
## 80 2021-08-01      NA    NA           NA              NA
## 81 2021-09-01      NA    NA           NA              NA
## 82 2021-10-01      NA    NA           NA              NA
## 83 2021-11-01      NA    NA           NA              NA
## 84 2021-12-01      NA    NA           NA              NA
## 85 2022-01-01      NA    NA           NA              NA
## 86 2022-02-01      NA    NA           NA              NA
## 87 2022-03-01      NA    NA           NA              NA
## 88 2022-04-01      NA    NA           NA              NA
## 89 2022-05-01      NA    NA           NA              NA
## 90 2022-06-01      NA    NA           NA              NA
## 91 2022-07-01      NA    NA           NA              NA
## 92 2022-08-01      NA    NA           NA              NA
## 93 2022-09-01      NA    NA           NA              NA
## 94 2022-10-01      NA    NA           NA              NA
## 95 2022-11-01      NA    NA           NA              NA
## 96 2022-12-01      NA    NA           NA              NA
```

A partir da linha 65, todos os valores são NA. Vamos removê-los.

```
data = drop_na(data)
```

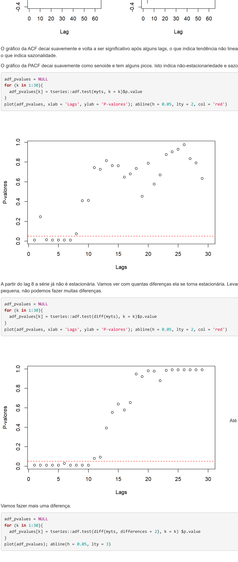
Análise primária da série temporal

```
myts = ts(data$Revenue, start = c(2015, 1), end = c(2020, 4), frequency = 12)
myts_decomp = decompose(myts)
autoplot(myts_decomp) # sazonalidade perceptível e não há estacionaredade
```

Aparentemente não há estacionaredade, a sazonalidade bem perceptível e os erros aparentemente são homocedasticos.

Análise dos gráficos de autocorrelação (ACF) e autocorrelação parcial (PACF)

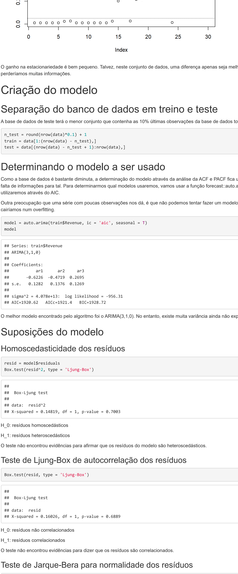
```
par(mfrow = c(1,2))
Acf(data$Revenue, lag.max = 96, main = 'ACF')
Pacf(data$Revenue, lag.max = 96, main = 'PACF')
```



O gráfico da ACF decai suavemente e volta a ser significativo após alguns lags, o que indica tendência não linear. Também contém alguns picos, o que indica sazonalidade.

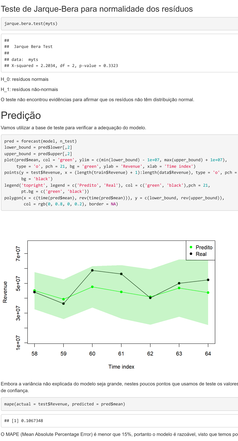
O gráfico da PACF decai rapidamente com sendo o tem alguns picos. Isto indica não-estacionaredade e sazonalidade.

```
adf_pvalues = NULL
for (k in 1:96){
  adf_pvalues[k] = tseries::adf.test(myts, k = k)$p.value
}
plot(adf_pvalues, xlab = 'Lags', ylab = 'P-valores'); abline(h = 0.05, lty = 2, col = 'red')
```



A partir do lag 8 a série já não é estacionária. Vamos ver com quantas diferenças ela se torna estacionária. Levando em conta que a série é pequena, eu poderia fazer muitas diferenças.

```
adf_pvalues = NULL
for (k in 1:96){
  adf_pvalues[k] = tseries::adf.test(diff(myts), k = k)$p.value
}
plot(adf_pvalues, xlab = 'Lags', ylab = 'P-valores'); abline(h = 0.05, lty = 2, col = 'red')
```



Até o lag 10 a série é estacionária.

Vamos fazer mais uma diferença.

```
adf_pvalues = NULL
for (k in 1:96){
  adf_pvalues[k] = tseries::adf.test(diff(myts, differences = 2), k = k) $p.value
}
plot(adf_pvalues); abline(h = 0.05, lty = 3)
```


O ganho na estacionaredade é bem pequeno. Talvez, neste conjunto de dados, uma diferença apenas seja melhor, pois com mais diferenças perderíamos muitas informações.

Criação do modelo

Separação do banco de dados em treino e teste

A base de dados de teste terá o menor conjunto que contenha as 10% últimas observações da base de dados total.

```
n_test = round(nrow(data)*0.1) + 1
train = data[1:(nrow(data) - n_test),]
test = data[(nrow(data) - n_test + 1):nrow(data),]
```

Determinando o modelo a ser usado

Como a base de dados é bastante diminuta, a determinação do modelo através da análise da ACF e PACF fica um pouco comprometida, pois há falta de informações para tal. Para determinarmos qual modelos usaremos, vamos usar a função forecast::auto.arima(), que decidirá qual modelo utilizaremos através do AIC.

Outra preocupação que uma série com poucas observações nos dá, é que não podemos tentar fazer um modelo complexo, pois com certeza cairíamos num overfitting.

```
model = auto.arima(train$Revenue, ic = 'aic', seasonal = T)
model
```

```
## Series: train$Revenue
## ARIMA(1,0,0)
##
## Coefficients:
##         ar1     ar2     ar3
##      0.3235  -0.0719  0.2695
## s.e.  0.1202   0.1376  0.1269
##
## sigma^2 = 8.478e+11:  log likelihood = -956.31
## AIC=1928.62   AICc=1929.4   BIC=1938.72
```

O melhor modelo encontrado pelo algoritmo foi o ARIMA(3,1,0). No entanto, existe muita variância ainda não explicada pelo modelo.

Suposições do modelo

Homocedasticidade dos resíduos

```
resid = model$residuals
Box.test(resid^2, type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  resid^2
## X-squared = 0.14819, df = 1, p-value = 0.7003
```

H_0: resíduos homocedasticos

H_1: resíduos heterocedasticos

O teste não encontrou evidências para afirmar que os resíduos do modelo são heterocedásticos.

Teste de Ljung-Box de autocorrelação dos resíduos

```
Box.test(resid, type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  resid
## X-squared = 0.16826, df = 1, p-value = 0.6889
```

H_0: resíduos não correlacionados

H_1: resíduos correlacionados

O teste não encontrou evidências para dizer que os resíduos são correlacionados.

Teste de Jarque-Bera para normalidade dos resíduos

```
jarque.bera.test(myts)
```

```
##
##  Jarque Bera Test
##
## data:  myts
## X-squared = 2.2854, df = 2, p-value = 0.3123
```

H_0: resíduos normais

H_1: resíduos não-normais

O teste não encontrou evidências para afirmar que os resíduos não têm distribuição normal.

Predição

Vamos utilizar a base de teste para verificar a adequação do modelo.

```
pred = forecast(model, n_test)
lower_bound = pred$lower[,2]
upper_bound = pred$upper[,2]
plot(pred$mean, col = 'green', ylim = c(min(lower_bound) - 1e+07, max(upper_bound) + 1e+07),
     type = 'b', pch = 21, bg = 'green', ylab = 'Revenue', xlab = 'Time Index')
points(x = test$Revenue, x = (length(train$Revenue) + 1):length(data$Revenue), type = 'o', pch = 22,
       bg = 'black')
legend('topright', legend = c('Predito', 'Real'), col = c('green', 'black'),pch = 21,
       pt.bg = c('green', 'black'))
polygon(x = c(time(pred$mean), rev(time(pred$mean))), y = c(lower_bound, rev(upper_bound)),
        col = rgb(0, 0.8, 0, 0.2), border = NA)
```


Embora a variância não explicada do modelo seja grande, nestes poucos pontos que usamos de teste os valores preditos não saíram do intervalo de confiança.

```
mape(actual = test$Revenue, predicted = pred$mean)
```

```
## [1] 0.3867348
```

O MAPE (Mean Absolute Percentage Error) é menor que 15%, portanto o modelo é razoável, visto que temos poucas observações.