REGRESION LINEAL MULTIPLE TOTALES DIARIOS

Miguel Angel Villegas

2025-03-10

*library(tsDyn)

*library(tidyverse)

*library(dplyr)

*library(readxl)

*library(EnvStats)

*library(corrplot)

*library(caTools)

library(GGally)

*library(forecast)

Introducción

Este modelo abarca todos las ventas de los productos, se utilizan las variables: Valor Unitario y Cantidad. Se realizan las correlaciones entre las tres variables y se determina el modelo lineal multiple. El conjunto de entrenamiento y prueba esta dividio en una proporcion de 80/20, sin embargo, la división es aleatoria,lo hace que el resultado sea mas confiable. Se siembra una semilla para permitir que los valores de la muestra sean los mismos.

```
ruta <- "/cloud/project/Ventas_Suministros_Totales.xlsx"
excel_sheets(ruta)
```

```
## [1] "Ventas Totales Original"    "Servicios Totales Original"
```

```
# "Ventas Totales Original"    "Servicios Totales Original"
```

```
Productos_Totales <- as.data.frame(read_xlsx(ruta,
                                             sheet = "Ventas Totales Original"))
Productos_Totales$Semana <- format(Productos_Totales$Fecha, format = "%Y-%U")
Productos_Totales$mes <- format(Productos_Totales$Fecha, format = "%Y-%m")

Servicios_Totales <- as.data.frame(read_xlsx(ruta,
                                             sheet = "Servicios Totales Original"))
Servicios_Totales$Semana <- format(Servicios_Totales$Fecha, format = "%Y-%U")
Servicios_Totales$mes <- format(Servicios_Totales$Fecha, format = "%Y-%m")

datatotal <- merge(x = Productos_Totales, y = Servicios_Totales, all = T)
colnames(datatotal) <- c("Indice", "Fecha", "RFC", "Empresa", "Cantidad",
                         "Pieza", "Descripcion", "ValorUnitario", "Total",
                         "Semana", "Mes" )
```

Se calcula el valor del parámetro lambda para la transformación Boxcox

```
RLM_dftotal_dia <- datatotal %>%
  group_by(Fecha = as.Date(Fecha)) %>%
  summarize(Ventas_Totales = sum(Total),
            Ventas_Unitario = sum(ValorUnitario),
            Ventas_Cantidad = sum(Cantidad),
            .groups = "keep")
head(RLM_dftotal_dia)
```

```
## # A tibble: 6 x 4
## # Groups:   Fecha [6]
##   Fecha      Ventas_Totales Ventas_Unitario Ventas_Cantidad
##   <date>              <dbl>           <dbl>           <dbl>
## 1 2019-07-01         46416.          20301.              88
## 2 2019-07-03         14100.            995               63
## 3 2019-07-04          5330.           1727.               7
## 4 2019-07-05         10146.           8746.               3
## 5 2019-07-06         10962             630               15
## 6 2019-07-08         16194.           3740               16
```

```
nrow(RLM_dftotal_dia)
```

```
## [1] 774
```

```
RLM_dftotal_sem <- as.data.frame(RLM_dftotal_dia)
head(RLM_dftotal_dia)
```

```
## # A tibble: 6 x 4
## # Groups:   Fecha [6]
##   Fecha      Ventas_Totales Ventas_Unitario Ventas_Cantidad
##   <date>              <dbl>           <dbl>           <dbl>
## 1 2019-07-01         46416.          20301.              88
## 2 2019-07-03         14100.            995               63
## 3 2019-07-04          5330.           1727.               7
## 4 2019-07-05         10146.           8746.               3
## 5 2019-07-06         10962             630               15
## 6 2019-07-08         16194.           3740               16
```

```
VT_lambda_dia <- boxcox(RLM_dftotal_dia$Ventas_Totales,
                        objective.name = "Log-Likelihood", optimize = T)
# 0.01997296
VU_lambda_dia <- boxcox(RLM_dftotal_dia$Ventas_Unitario,
                        objective.name = "Log-Likelihood", optimize = T)
# 0.07191032
VC_lambda_dia <- boxcox(RLM_dftotal_dia$Ventas_Cantidad,
                        objective.name = "Log-Likelihood", optimize = T)
#-0.2234713
```

Se obtiene la transformación boxcox para las ventas totales, valor unitario de los productos y cantidad de venta de los productos, todos por dia

```
RLM_dftotal_dia <- RLM_dftotal_dia %>%
  mutate(
    Ventas_Totales = boxcoxTransform(Ventas_Totales, lambda = 0.01997296),
    Ventas_Unitario = boxcoxTransform(Ventas_Unitario, lambda = 0.07191032),
    Ventas_Cantidad = boxcoxTransform(Ventas_Cantidad, lambda = -0.2234713)
```

```
  )
head(RLM_dftotal_dia)
```

```
## # A tibble: 6 x 4
## # Groups:   Fecha [6]
##   Fecha      Ventas_Totales Ventas_Unitario Ventas_Cantidad
##   <date>              <dbl>           <dbl>           <dbl>
## 1 2019-07-01          12.0            14.5            2.83
## 2 2019-07-03          10.5             8.94           2.70
## 3 2019-07-04           9.36            9.86           1.58
## 4 2019-07-05          10.1            12.8            0.974
## 5 2019-07-06          10.2             8.20           2.03
## 6 2019-07-08          10.7            11.2            2.07
```

Se seleccionan las columnas que son de interes

```
## Adding missing grouping variables: `Fecha`
```

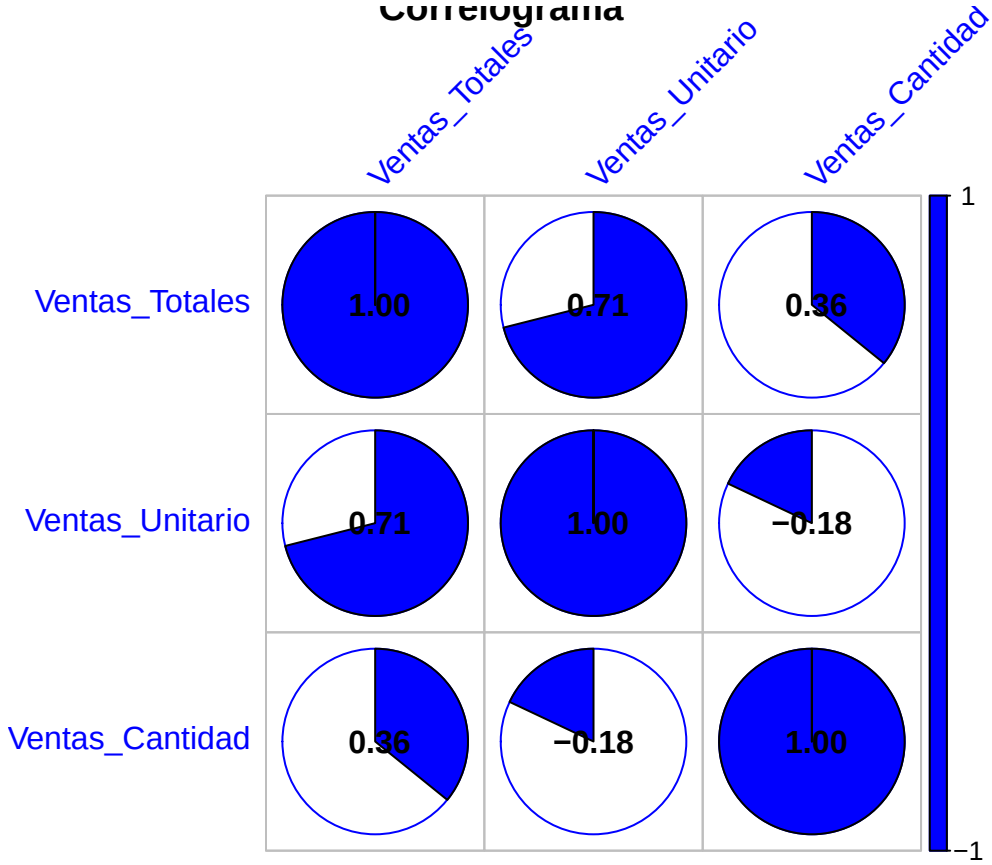
Se crea una matriz para el cálculo y visualización de las correlaciones, además se siembra la semilla para garantizar quelos valores sean los mismos.

```
set.seed(101)

RLM_dftotal_dia_mtx <- cbind(RLM_dftotal_dia$Ventas_Totales,
                             RLM_dftotal_dia$Ventas_Unitario,
                             RLM_dftotal_dia$Ventas_Cantidad)
colnames(RLM_dftotal_dia_mtx) <- c("Ventas_Totales", "Ventas_Unitario", "Ventas_Cantidad")
head(RLM_dftotal_dia_mtx)
```

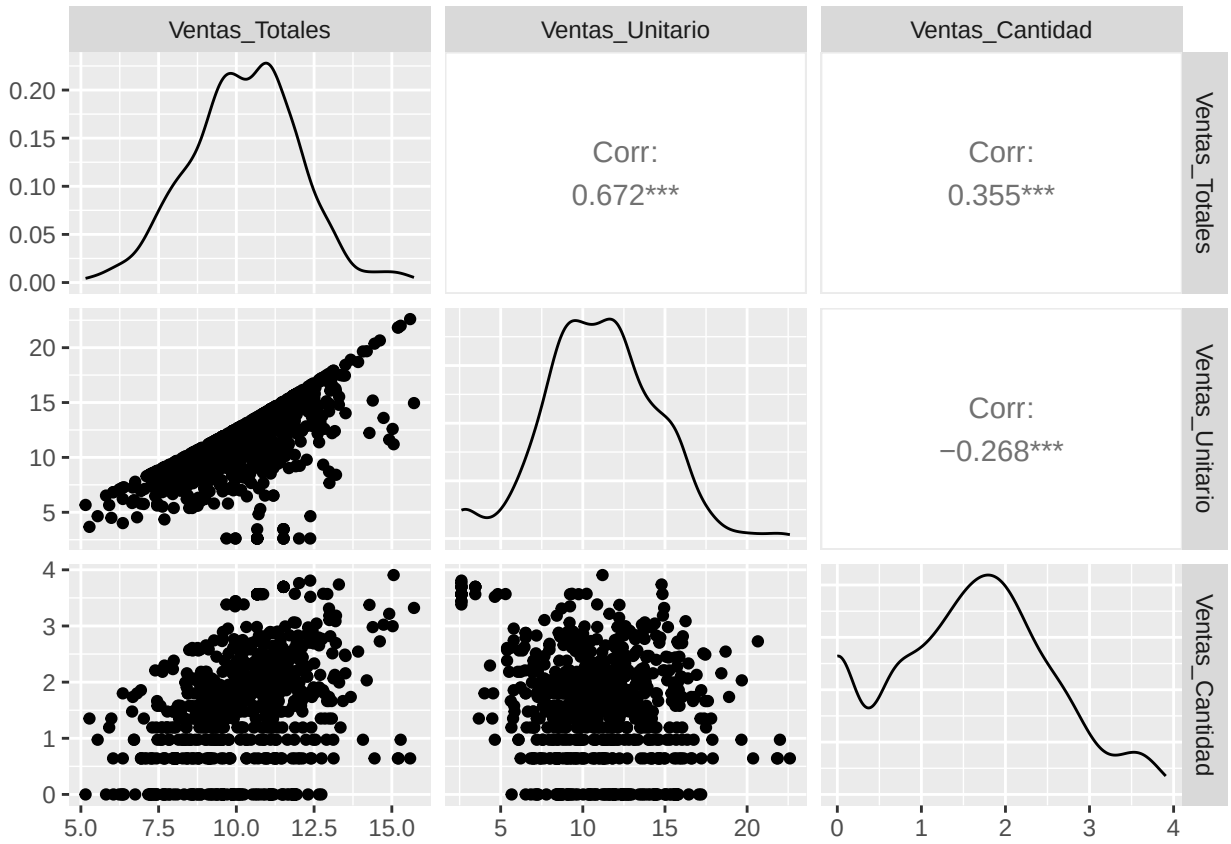
```
##      Ventas_Totales Ventas_Unitario Ventas_Cantidad
## [1,]      11.985596       14.470619        2.829559
## [2,]      10.526310        8.938371        2.701976
## [3,]       9.360276        9.862672        1.578017
## [4,]      10.129293       12.803311        0.974144
## [5,]      10.222425        8.199795        2.031667
## [6,]      10.694106       11.220471        2.066651
```

Gráficas de correlación

```
##      Ventas_Totales Ventas_Unitario Ventas_Cantidad
## [1,]      11.985596       14.470619        2.829559
## [2,]      10.526310        8.938371        2.701976
## [3,]       9.360276        9.862672        1.578017
## [4,]      10.129293       12.803311        0.974144
## [5,]      10.222425        8.199795        2.031667
## [6,]      10.694106       11.220471        2.066651
```




```
RLM_dftotal_dia <- as.data.frame(RLM_dftotal_dia_mtx)
RLM_dftotal_dia %>% GGally::ggpairs(cardinality_threshold = 10)
```



Se hace la division del conjunto de datos en una proporcion de 80-20

```
m_lm <- sample.split(RLM_dftotal_dia$Ventas_Totales, SplitRatio = 0.80)
e_lm <- subset(RLM_dftotal_dia, m_lm == T)
p_lm <- subset(RLM_dftotal_dia, m_lm == F)
```

Modelo

Se determina el modelo lineal

```
Molm <- lm(Ventas_Totales ~ ., data = e_lm)
```

Resumen del modelo obtenido

```
print(summary(Molm))
```

```
##
## Call:
## lm(formula = Ventas_Totales ~ ., data = e_lm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.21177 -0.60838 -0.09866  0.49476  3.04760
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.09178    0.14611   28.00   <2e-16 ***
## Ventas_Unitario  0.41936    0.01039   40.37   <2e-16 ***
## Ventas_Cantidad  1.00520    0.03593   27.98   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8457 on 616 degrees of freedom
## Multiple R-squared:  0.7582, Adjusted R-squared:  0.7574
## F-statistic: 965.9 on 2 and 616 DF,  p-value: < 2.2e-16
```

```
# Multiple R-squared:  0.7582,  Adjusted R-squared:  0.7574
```

Pronostico

```
pronostico_lm <- predict(Molm, p_lm)
```

Se crea una data frame con los resultados y los valores actuales

```
resultados <- cbind(pronostico_lm, p_lm$Ventas_Totales)
resultados <- as.data.frame(resultados)
colnames(resultados) <- c("prediccion", "actual")
head(resultados)
```

```
##    prediccion    actual
## 11  10.964983  9.877478
## 14  10.742000 10.643535
## 17  11.304470 11.141824
## 22   9.134216  8.524057
## 25  10.740348 11.018052
## 26   9.432301  9.954320
```

Si es hay valores menores que cero se subtituyen por cero

```
any(resultados < 0)
```

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

Funcion

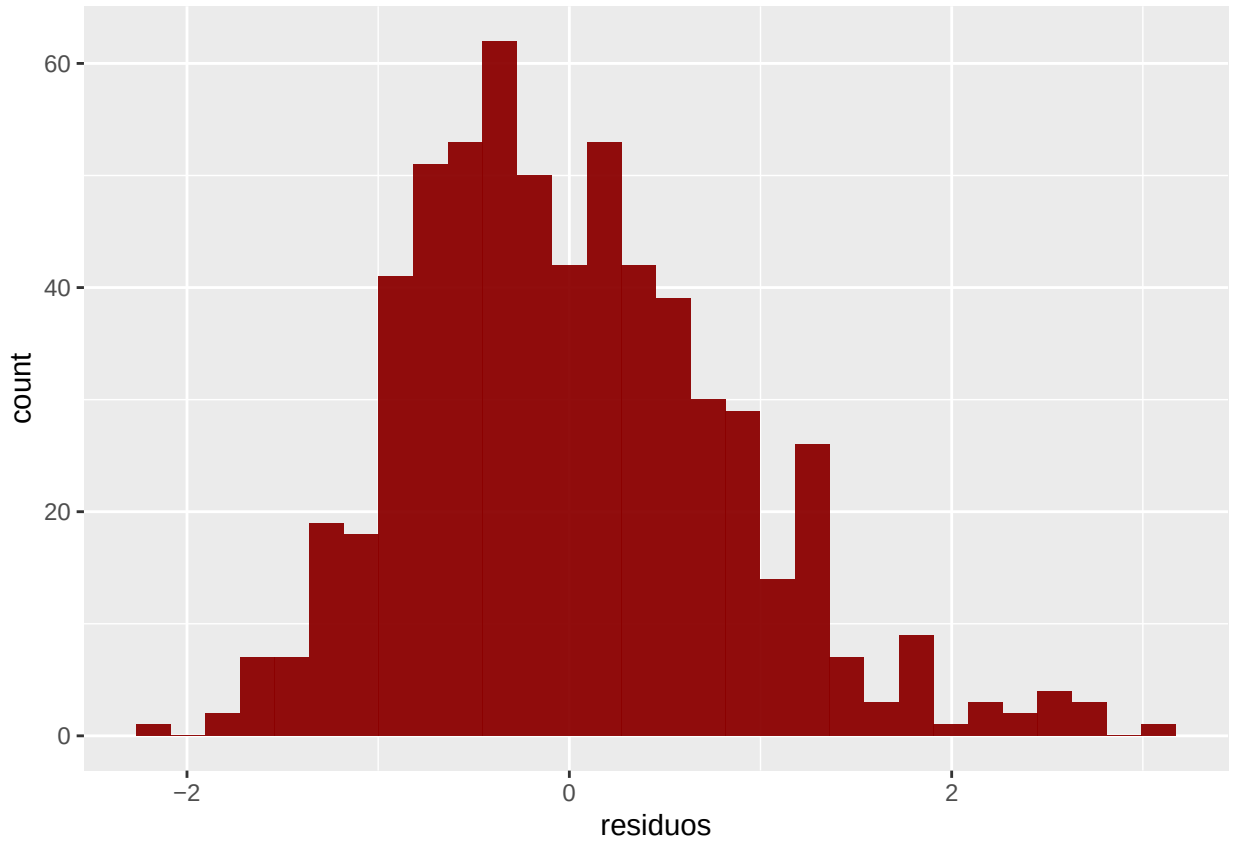
Exactitud del modelo

```
summary(Molm)$r.squared
```

```
## [1] 0.7582135
```

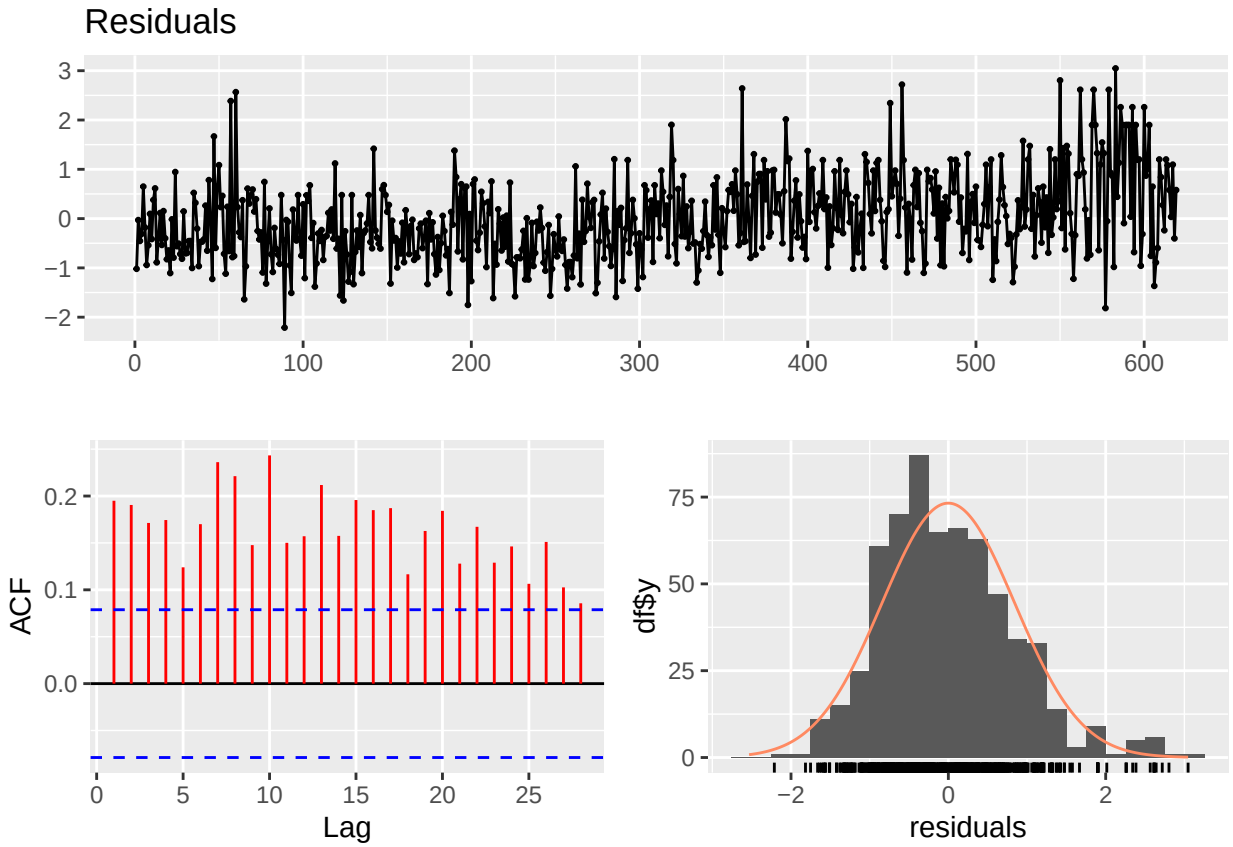
```
# [1] 0.7582135
```

Inspección de los residuales



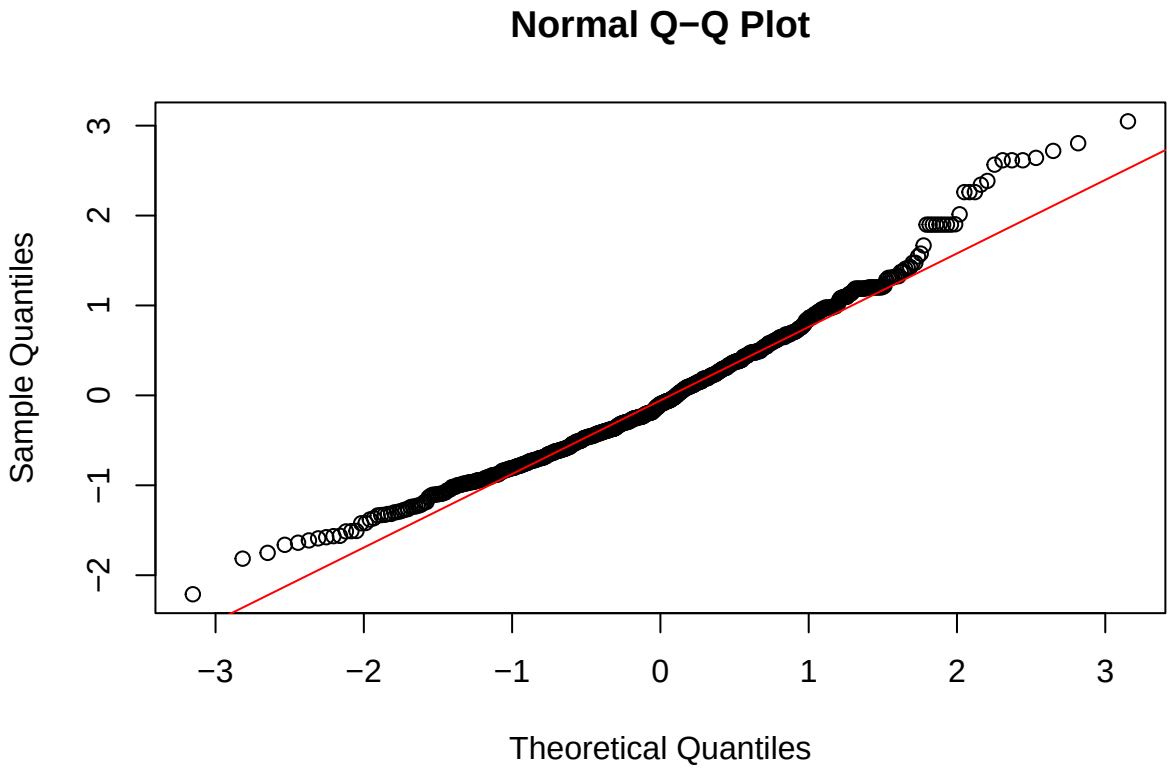
Residuales

```
checkresiduals(Molm, col = "red")
```




```
##
##  Breusch-Godfrey test for serial correlation of order up to 10
##
## data:  Residuals
## LM test = 94.806, df = 10, p-value = 5.937e-16
```

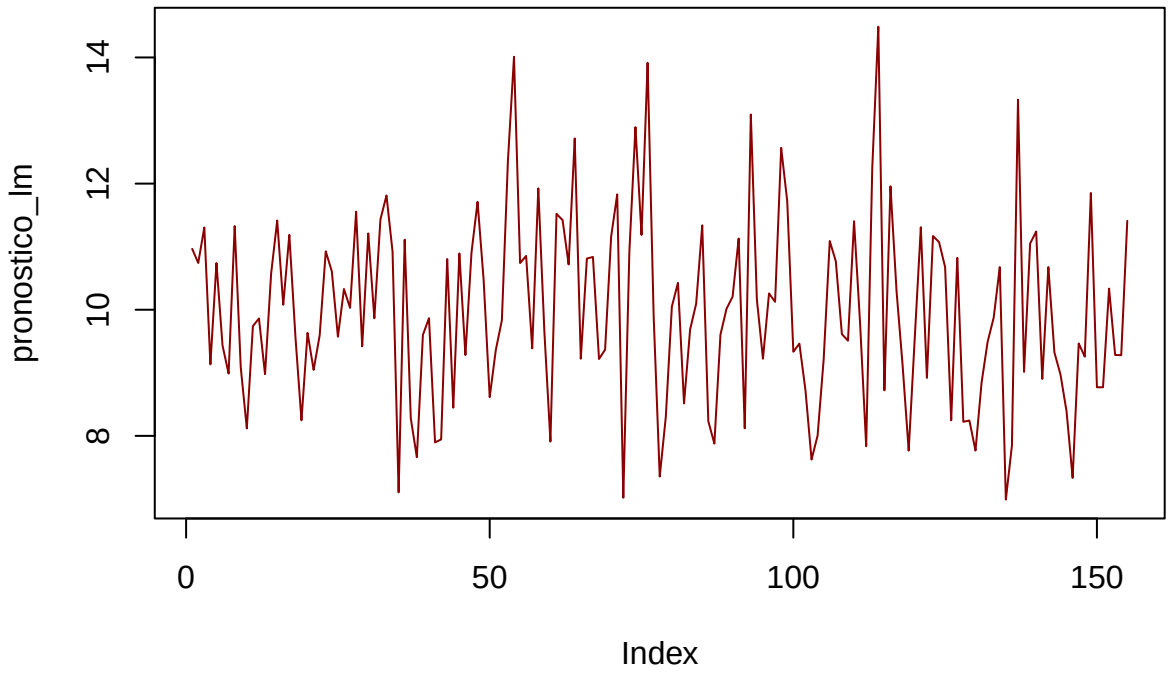
Inspeccionando si existe normalidad en los residuales



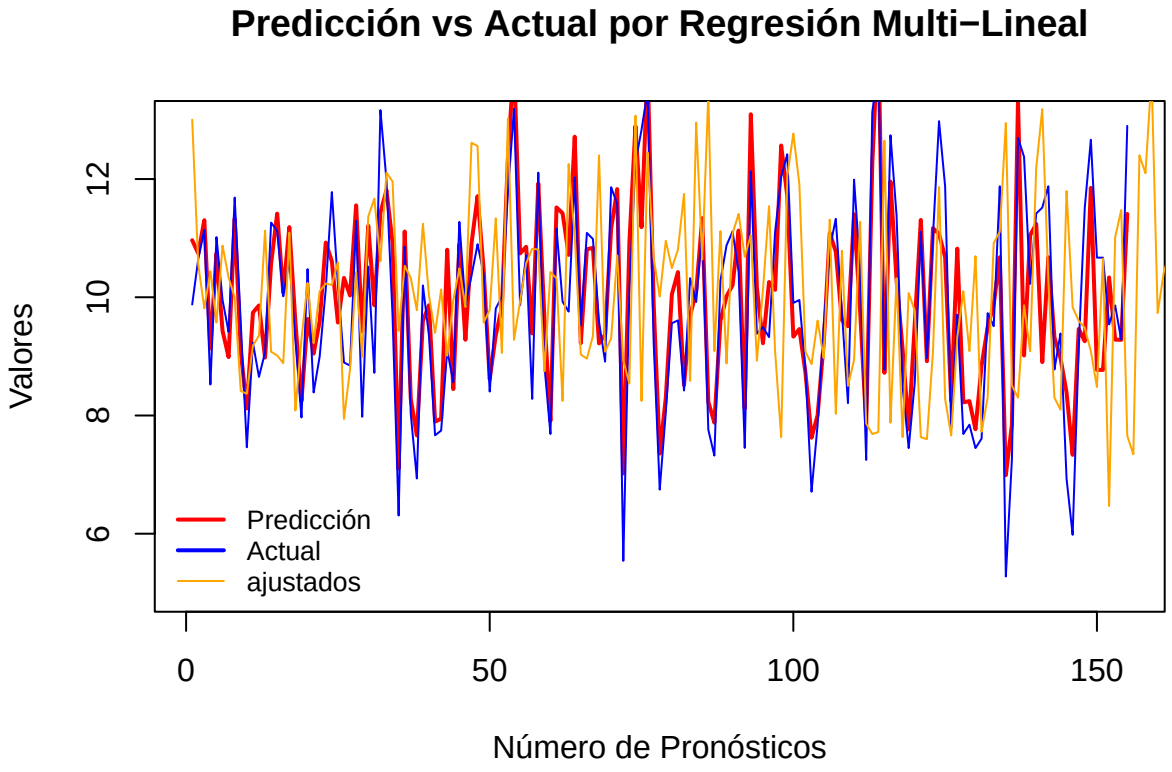
Se grafican los pronosticos

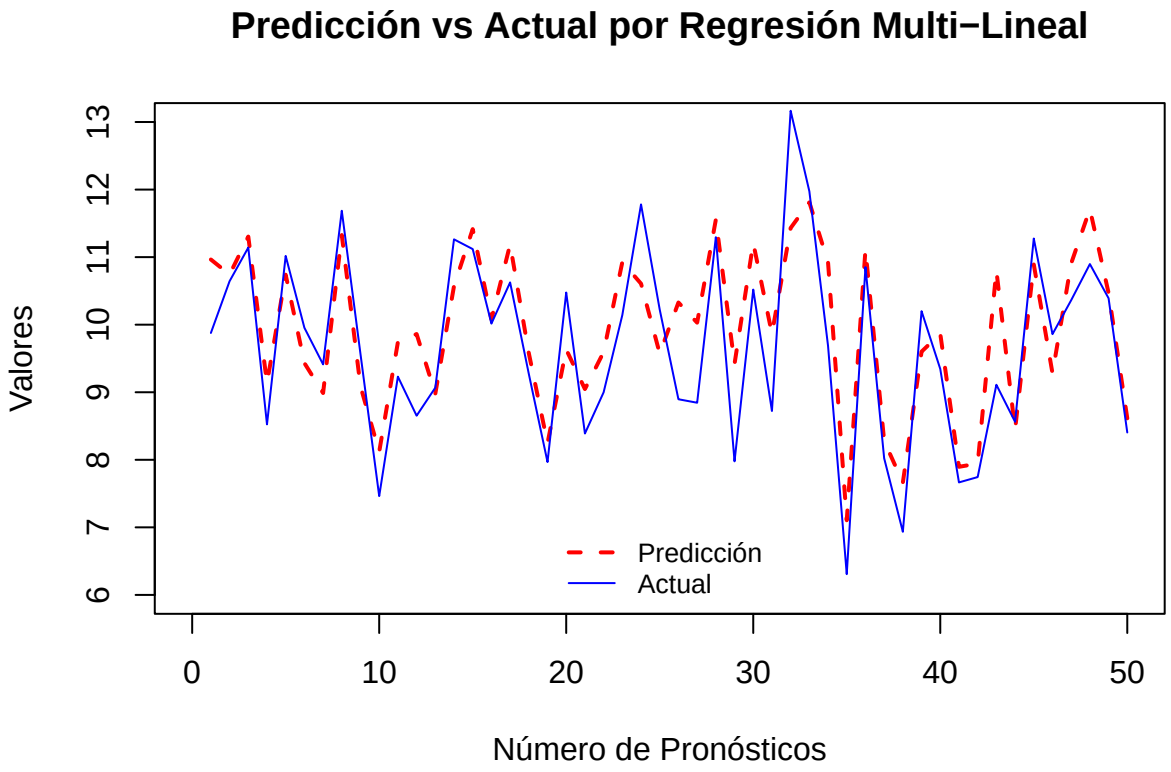
```
pronostico_lm <- predict(Molm, p_lm)

plot(pronostico_lm, type = "lines", col = "darkred", yalab = "Valores")
```

```
## Warning in plot.window(...): "yalab" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): plot type 'lines' will be truncated to first
## character

## Warning in plot.xy(xy, type, ...): "yalab" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "yalab" is not a
## graphical parameter
## Warning in axis(side = side, at = at, labels = labels, ...): "yalab" is not a
## graphical parameter

## Warning in box(...): "yalab" is not a graphical parameter

## Warning in title(...): "yalab" is not a graphical parameter
```



Gráfica de los pronosticos junto con los valores reales





Se determina la exactitud del modelo

```
accuracy(resultados$prediccion[1:50], p_lm$Ventas_Totales[1:50])
```

```
##                      ME      RMSE       MAE       MPE     MAPE
## Test set -0.2503733 0.7477632 0.6136234 -3.206985 6.520881
```

```
#                      ME      RMSE       MAE       MPE     MAPE
# Test set -0.2503733 0.7477632 0.6136234 -3.206985 6.520881
```

Conclusiones

El modelo captura 75.82% la dinamica de la serie, sin embargo, los residuales estan muy por debajo del valor ideal de $p > 0.05$, lo que indica que existe una fuerte correlacion entre los residuos. Si bien captura la dinamica de los valores, la correlación residual puede presentar un problema para ser considerado un modelo ideal de pronostico.