**Виконали: Кузьменко Юрій, Болотов Єгор**

## Лабораторна робота №8

# Модельна діагностика.

## Опис dataset

**Назва dataset:**

Spotify Top 10000 Streamed Songs

**Link на dataset:**

https://www.kaggle.com/datasets/rakkesharv/spotify-top-10000-streamed-songs

**Опис dataset та постановку задачі:**

Це набір даних, зібраний з веб-сайту Spotify, котрий містить потоки виконавця та кількість прослiховувань (було взято саме топ-10000) Основна мета: вплив факторів на популярність пісні й дізнатись найпопулярніших виконавців та треки.

**Змінні та їх опис:**

Position - Spotify Ranking

Artist Name - Artist Name

Song Name - Song Name

Days - No of days since the release of the song

Top 10 (xTimes) - No of times inside top 10

Peak Position - Peak position attained

Peak Position (xTimes) - No of times Peak position attained

Peak Streams - Total no of streams during Peak position

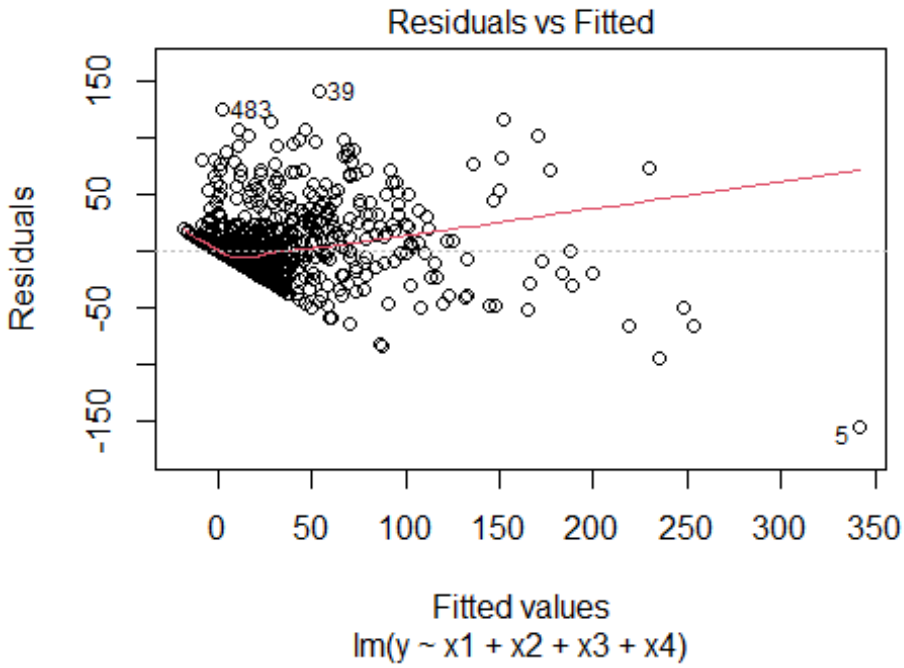
Total Streams - Total song streams

```r
y <- df$Top_ten_times
x1 <- df$Peak_position_times
x2 <- df$Peak_streams
x3 <- df$Days
x4 <- df$Total_streams
x5 <- df$Position
x6 <- df$Peak_position
mod <- lm(y ~ x1 + x2 + x3 + x4);
mod_1 <- lm(y ~ x1 + x2 + x3);


summary(mod)
## Call:
## lm(formula = y ~ x1 + x2 + x3 + x4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -155.584   -0.171   -0.039    0.073  140.212
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.315e-01  1.115e-01   2.972  0.00297 **
## x1           1.548e+00  2.733e-02  56.648  < 2e-16 ***
## x2          -1.175e-06  1.465e-07  -8.021 1.16e-15 ***
## x3          -6.552e-02  1.837e-03 -35.672  < 2e-16 ***
## x4           3.238e-07  5.166e-09  62.682  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.219 on 11079 degrees of freedom
## Multiple R-squared:  0.7227, Adjusted R-squared:  0.7226
## F-statistic:  7218 on 4 and 11079 DF,  p-value: < 2.2e-16

summary(mod_1)
## Call:
## lm(formula = y ~ x1 + x2 + x3)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -166.960   -1.038    0.602    1.089  166.111
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.786e+00  1.237e-01  -14.44   <2e-16 ***
## x1           2.390e+00  2.771e-02   86.26   <2e-16 ***
## x2           2.412e-06  1.569e-07   15.37   <2e-16 ***
## x3           4.200e-02  7.641e-04   54.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.565 on 11080 degrees of freedom
## Multiple R-squared:  0.6243, Adjusted R-squared:  0.6242
## F-statistic:  6138 on 3 and 11080 DF,  p-value: < 2.2e-16
```

## Завдання 1: Перевірити дані на лінійність

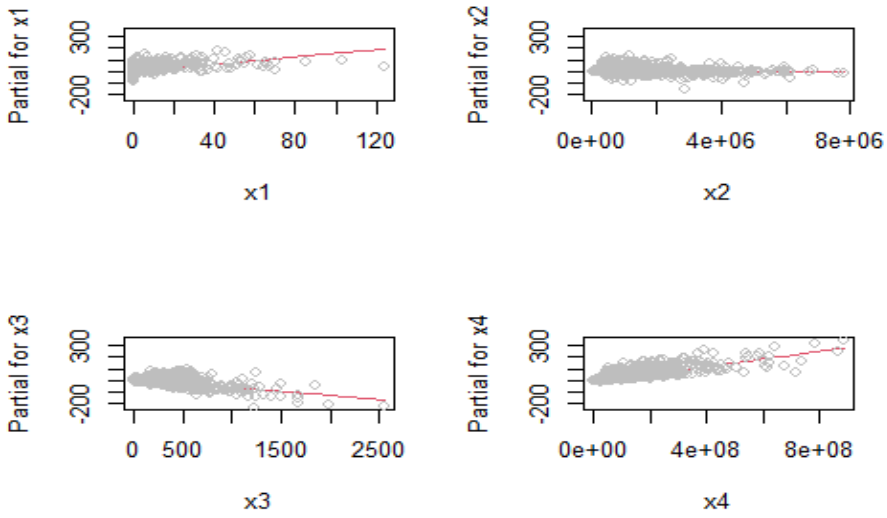
**(A)** Представити графічно plot(*, 1);

```
plot(mod,1)
```




**(B)** Побудувати залежності $Y \sim xi$, $i = 1, \ldots, 4$ termplot(mod, partial.resid = TRUE)

```
par(mfrow = c(2, 2))
termplot(mod, partial.resid = TRUE)
```

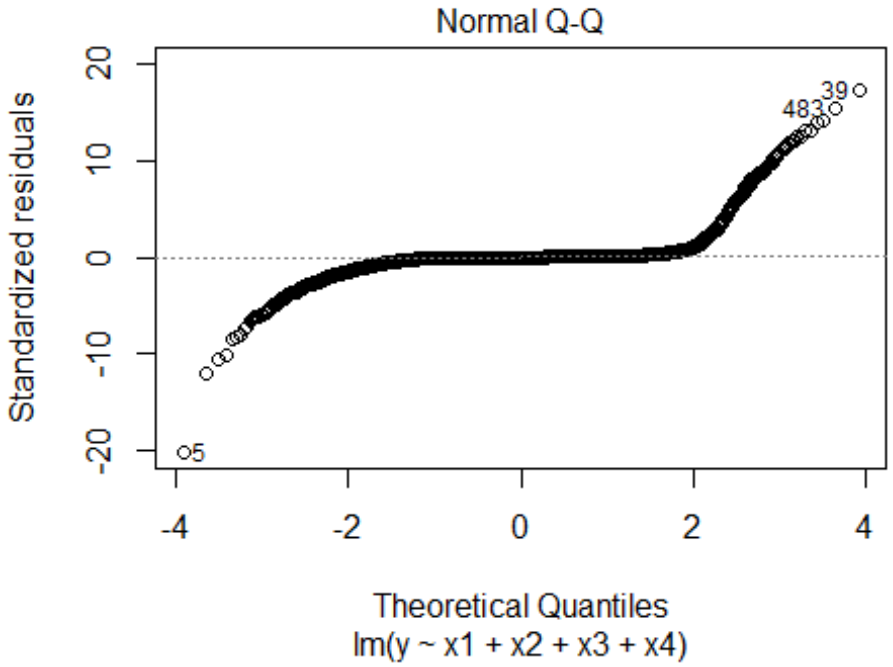


**(C)** Вказати які змінні потребують корекції;
**Найбільше потребує зміна змінної x1**

## Завдання 2: Перевірити дані на нормальність

**(A)** Представити графічно plot(*, 2);

```
plot(mod,2)
```




**(B)** тест Шапіро-Вілька

```
new_df <- df[sample(nrow(df), nrow(df)*0.45),]
ny <- new_df$Top_ten_times
nx1 <- new_df$Peak_position_times
nx2 <- new_df$Peak_streams
nx3 <- new_df$Days
nx4 <- new_df$Total_streams

nmod <- lm(ny ~ nx1 + nx2 + nx3 + nx4);

shapiro.test(nmod$residuals)

##
##  Shapiro-Wilk normality test
##
## data:  nmod$residuals
## W = 0.34363, p-value < 2.2e-16
```

Через обмеження у розмірі вибірки прийшлось зменшити розмір до 45% від загального розміру.

**(C)** тест Лілліфорса

```
nortest::lillie.test(mod$residuals)
```

```
##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  mod$residuals
## D = 0.35478, p-value < 2.2e-16
```

**(D)** Виконати перетворення Yeo-Johnson для змінної, яка *xi* яка на ваш погляд не є нормально розподілена

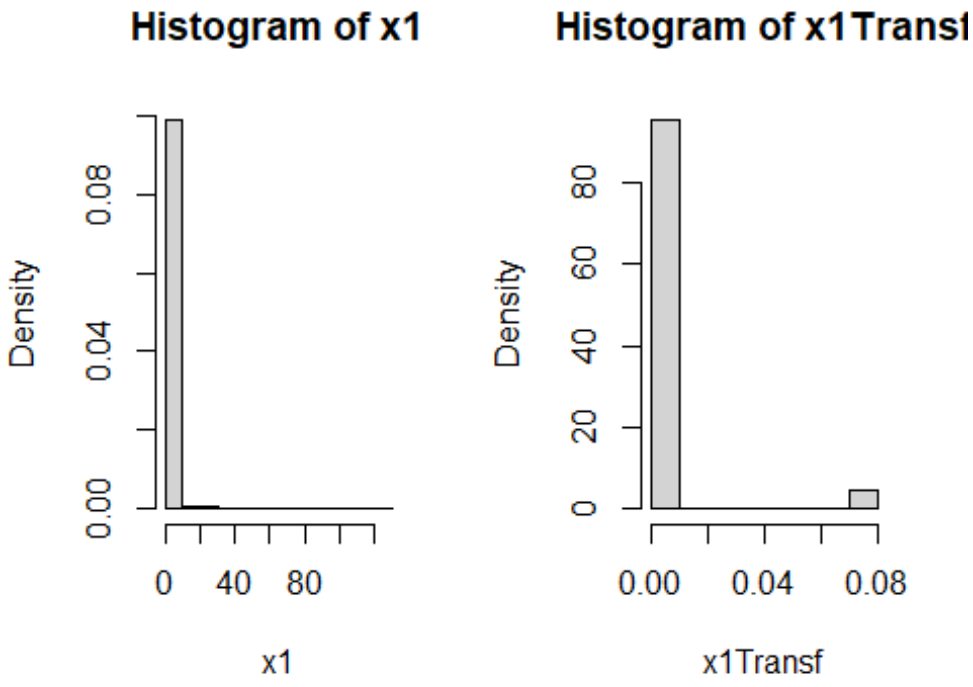
```
YJ <- car::powerTransform(lm(x1 ~ 1), family = "yjPower")
(lambdaYJ <- YJ$lambda)
```

```
##        Y1
## -13.95035
```

```
x1Transf <- car::yjPower(U = x1, lambda = lambdaYJ)
```

**(E)** Побудуйте гістограми для *xi* до трансформації і після;

```
par(mfrow = c(1, 2))
hist(x1, freq = FALSE, breaks = 10)
hist(x1Transf, freq = FALSE, breaks = 10)
```

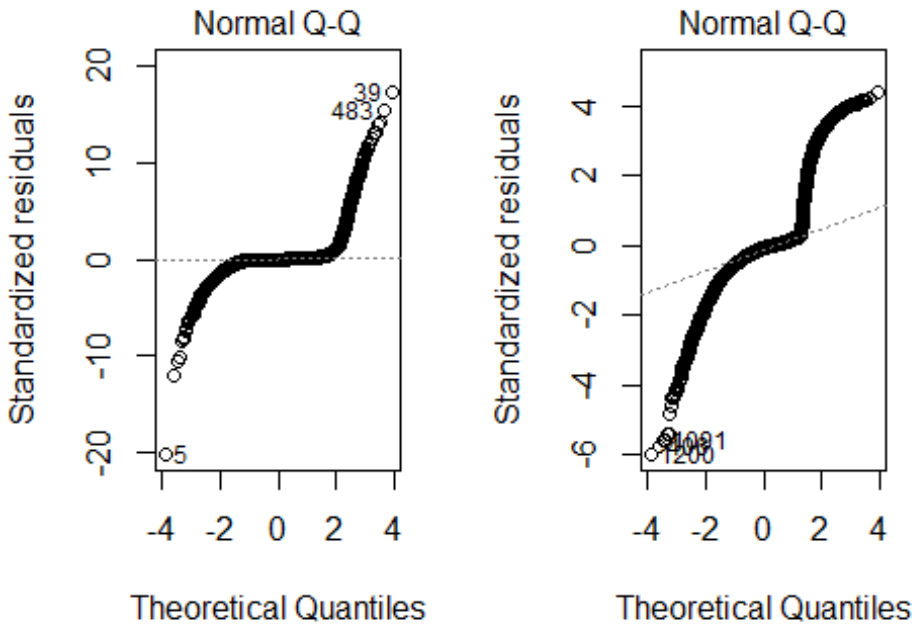



**(F)** Виконайте твансформацію Yeo-Johnson для Y (YTransf);

```
YTransf <- car::yjPower(U = y, lambda = lambdaYJ)
```

**(G)** Порівняйте plot(*, 2) для Y та YTransf

```
par(mfrow = c(1, 2))
plot(lm(y ~ x1 + x2 + x3 + x4), 2)
plot(lm(YTransf ~ x1 + x2 + x3 + x4),2)
```

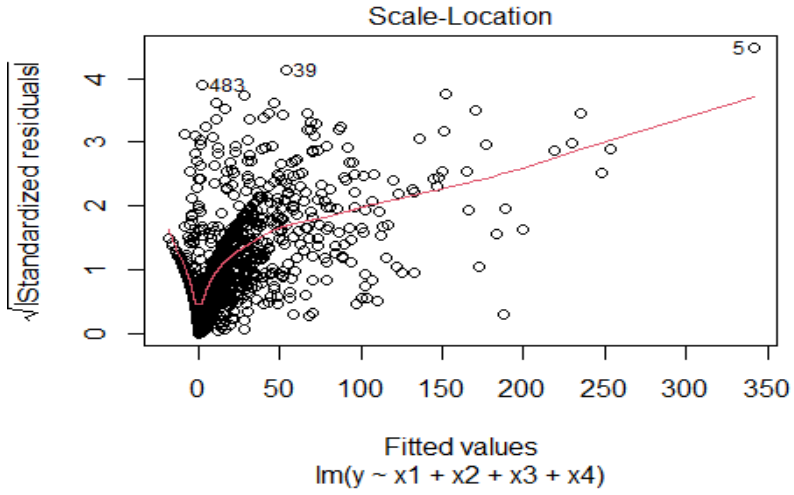



Дійсно графік став трохи кращим.

## Завдання 3: Перевірити дані на гомоскедастичність

**(A)** Представити графічно plot(*, 3);

```
plot(mod, 3)
```



**(B)** Для перевірки використати тест Брейша – Пагана;

```
car::ncvTest(mod)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 118048.2, Df = 1, p = < 2.22e-16
```

Гомоскедатичність відкидуємо

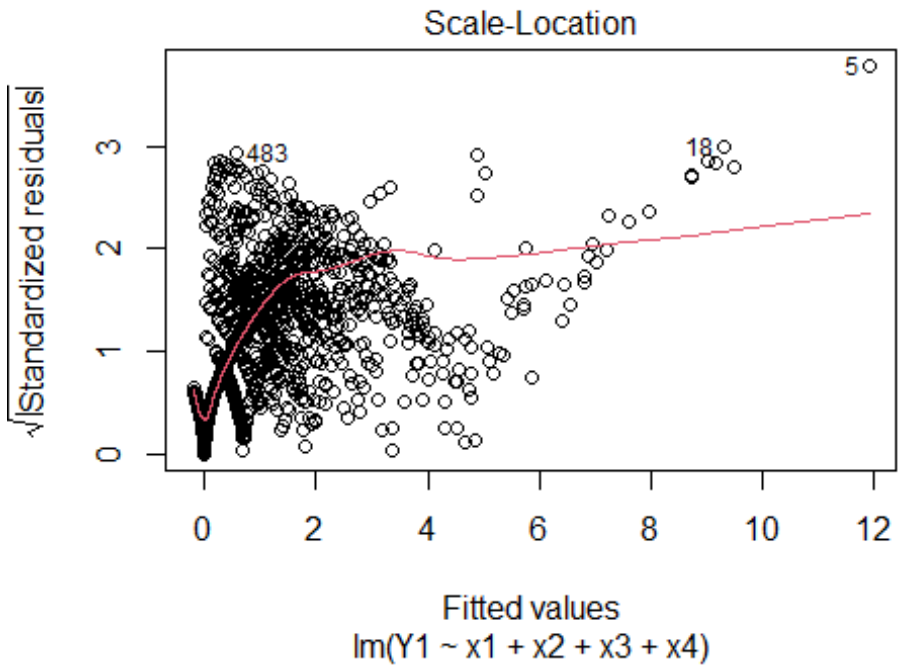
**(C)** Виконати перетворення для залежної змінної та перевірки за тестом Брейша – Пагана:

o Y1 <- log(abs(Y));

```
Y1 <- log(abs(y))
# У Y1 вийшло багато -inf -> замінюємо на 0
Y1[Y1 == -Inf] <- 0
mod_log_1 <- lm(Y1 ~ x1 + x2 + x3 + x4)
car::ncvTest(mod_log_1)
```

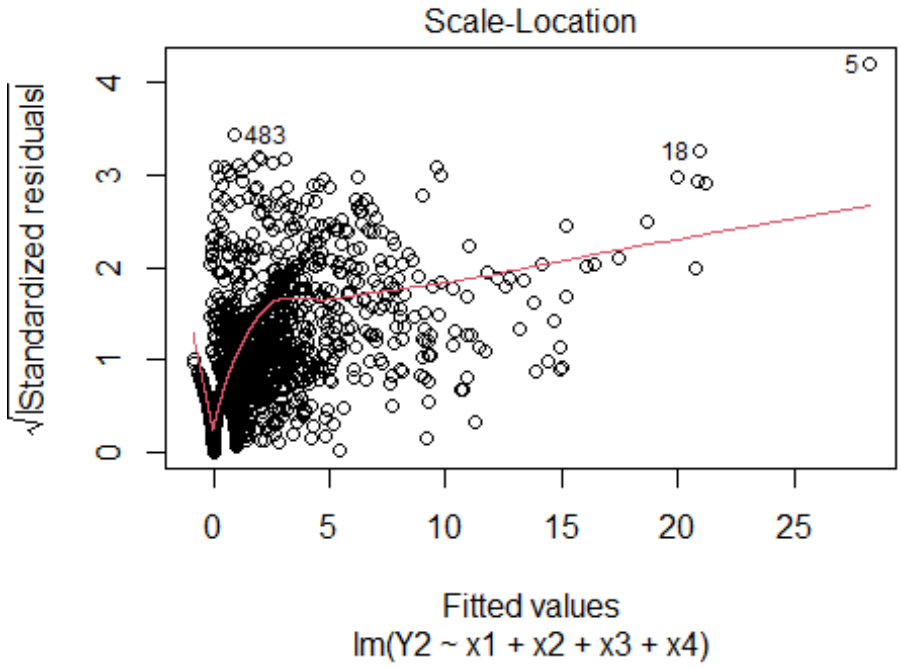
```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 32193.8, Df = 1, p = < 2.22e-16
```

```
plot(mod_log_1, 3)
```



o Y2 <- sqrt(abs(Y));

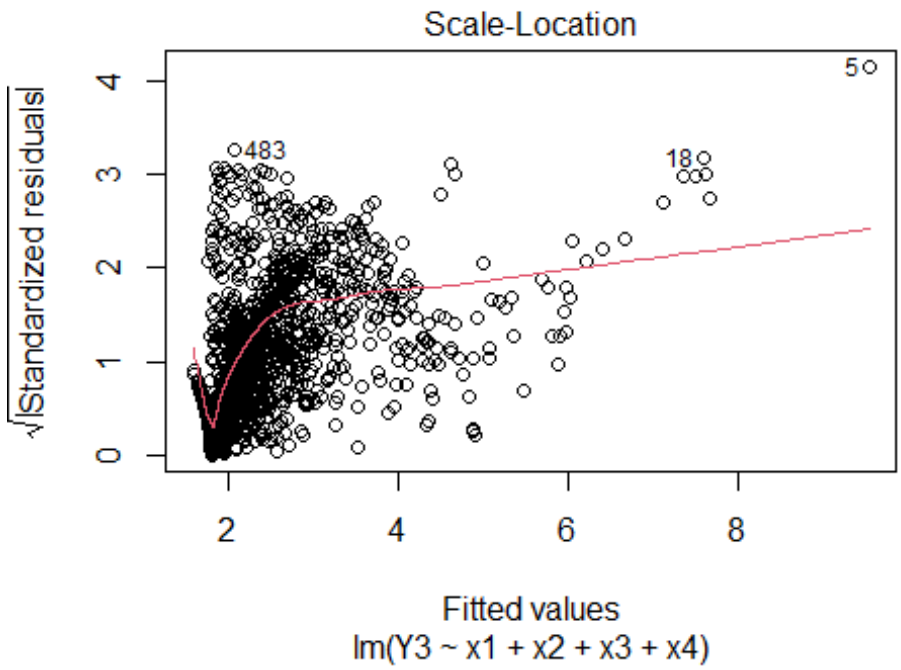
```
Y2 <- sqrt(abs(y))
mod_sqrt_1 <- lm(Y2 ~ x1 + x2 + x3 + x4)
car::ncvTest(mod_sqrt_1)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 48279.55, Df = 1, p = < 2.22e-16

plot(mod_sqrt_1, 3)
```




Краще не стало, багато розспипано по проміжку

o Трансформація Бокса-Кокса Y3 <- log(Y + m));

```
delta <- 6
m <- -min(y) + delta
Y3 <- I(log(y + m))
mod_log2_1 <- lm(Y3 ~ x1 + x2 + x3 + x4)
car::ncvTest(mod_log2_1)

## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 47367.64, Df = 1, p = < 2.22e-16

plot(mod_log2_1, 3)
```




Не те що нам потрібно, значення розсписані по проміжку

о Трансформація за Йо-Джонсоном Y4;

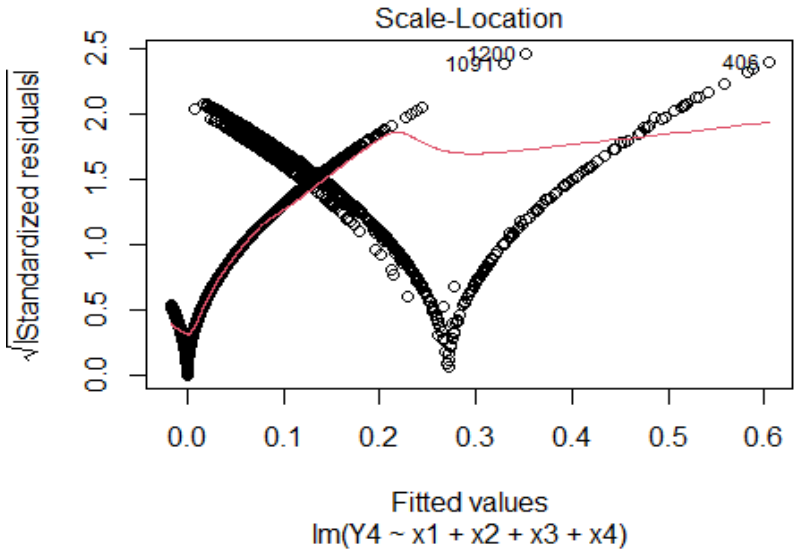
```
YJ <- car::powerTransform(lm(y ~ x1 + x2 + x3 + x4), family = "yjPower")
(lambdaYJ <- YJ$lambda)
```

```
##        Y1
## -3.691625
```

```
Y4 <- car::yjPower(U = y, lambda = lambdaYJ)
mod_YJ_1<- lm(Y4 ~ x1 + x2 +x3 + x4)
car::ncvTest(mod_YJ_1)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 10934.8, Df = 1, p = < 2.22e-16
```

```
plot(mod_YJ_1, 3)
```



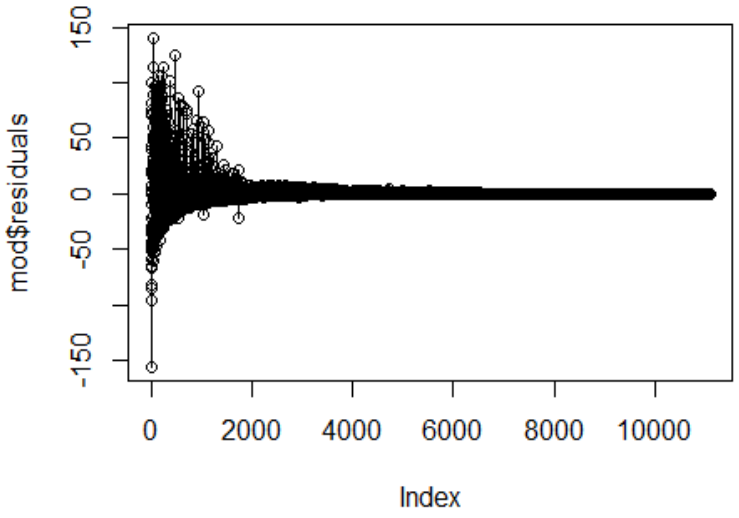

Пташка. Но все ж не схоже на те, що потрібно нам (значення на проміжку від [..0.0; ~0.25]) сходиться з лінією

В фінальному висновку до цього завдання можна вказати, що найкраще показала себе YJ трансформація

## Завдання 4: Перевірити дані на незалежність.

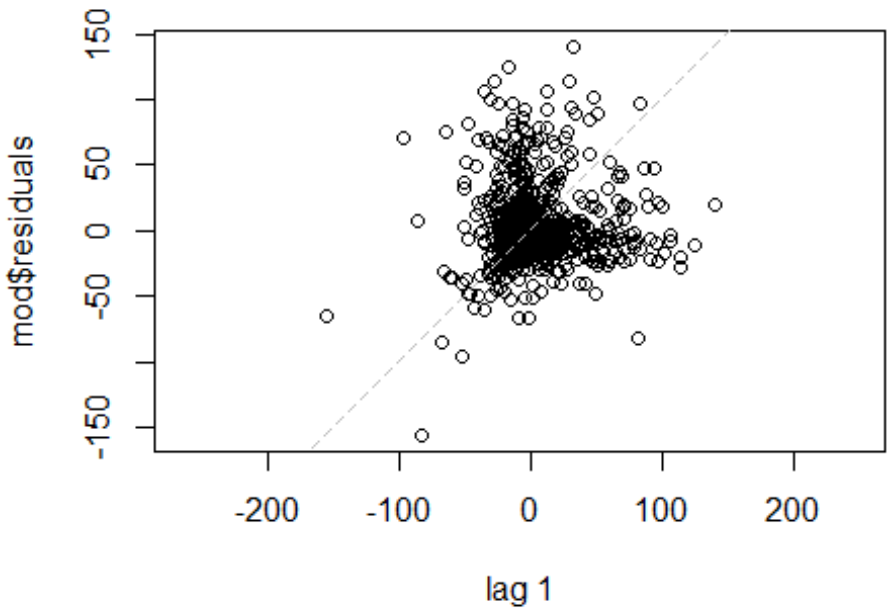
**(A)** Представити графічно та перевірити за plot(*$residuals, type = “o”);

```
plot(mod$residuals, type = "o")
```



**(B)** перевірити за lag.plot(*$residuals, lags = 1, do.lines = FALSE)

```
lag.plot(mod$residuals, lags = 1, do.lines = FALSE)
```




**(C)** перевірити за cor(*$residuals[-1], mod$residuals[-length(mod$residuals)])

```
cor(mod$residuals[-1], mod$residuals[-length(mod$residuals)])
```

```
## [1] 0.09364348
```

Значення кореляції невід'ємне, але мале

**(D)** Для перевірки використати тест Дарбіна – Ватсона;

```
car::durbinWatsonTest(mod)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.09331983      1.806461       0
##  Alternative hypothesis: rho != 0
```

## Завдання 5: Перевірити дані на мультиколінеарність.

**(A)** Представити залежність даних таблично round(cor(data), 2)

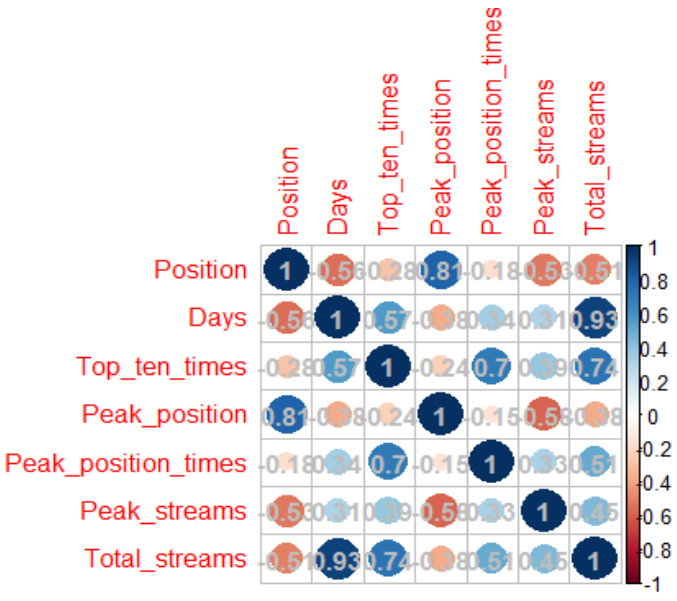
```
round(cor(df[, c(1, 4:9)]), 2)
```

```
##                     Position  Days Top_ten_times Peak_position
## Position                1.00 -0.56         -0.28          0.81
## Days                   -0.56  1.00          0.57         -0.38
## Top_ten_times          -0.28  0.57          1.00         -0.24
## Peak_position           0.81 -0.38         -0.24          1.00
## Peak_position_times    -0.18  0.34          0.70         -0.15
## Peak_streams           -0.53  0.31          0.39         -0.58
## Total_streams          -0.51  0.93          0.74         -0.38
##                     Peak_position_times Peak_streams Total_streams
## Position                          -0.18        -0.53         -0.51
## Days                               0.34         0.31          0.93
## Top_ten_times                      0.70         0.39          0.74
## Peak_position                     -0.15        -0.58         -0.38
## Peak_position_times                1.00         0.33          0.51
## Peak_streams                       0.33         1.00          0.45
## Total_streams                      0.51         0.45          1.00
```

Position залежить від Peak_Position, Days від Total_Streams, Peak_Position_times від Top_Ten_Times.

**(B)** Графічно corrplot::corrplot(cor(data), addCoef.col = “grey”);

```
corrplot::corrplot(cor(df[, c(1, 4:9)]), addCoef.col = "grey")
```




“Чому не використовуються всі змінні?”, на це є відповідь. Artist_Name та Song_Name є символьним типом й більшість з них унікальні (99%), тому їх можна відкинути (до того ж їх не використати у cor й тп) Залежні змінні були вказані у минулому задванні.

**(C)** Обчислити коефіцієнт Variance Inflation Factor (VIF) для моделі mod;

```
vif(mod)
```

```
##        x1        x2        x3        x4
##  1.592354  1.389861  9.318011 11.930474
```

**(D)** Обчислити коефіцієнт VIF в моделі mod_1;

```
vif(mod_1)
```

```
##       x1       x2       x3
## 1.208038 1.177744 1.190703
```

**(E)** Порівняти моделі за car::compareCoefs(mod, mod_1)

```
car::compareCoefs(mod, mod_1)
```

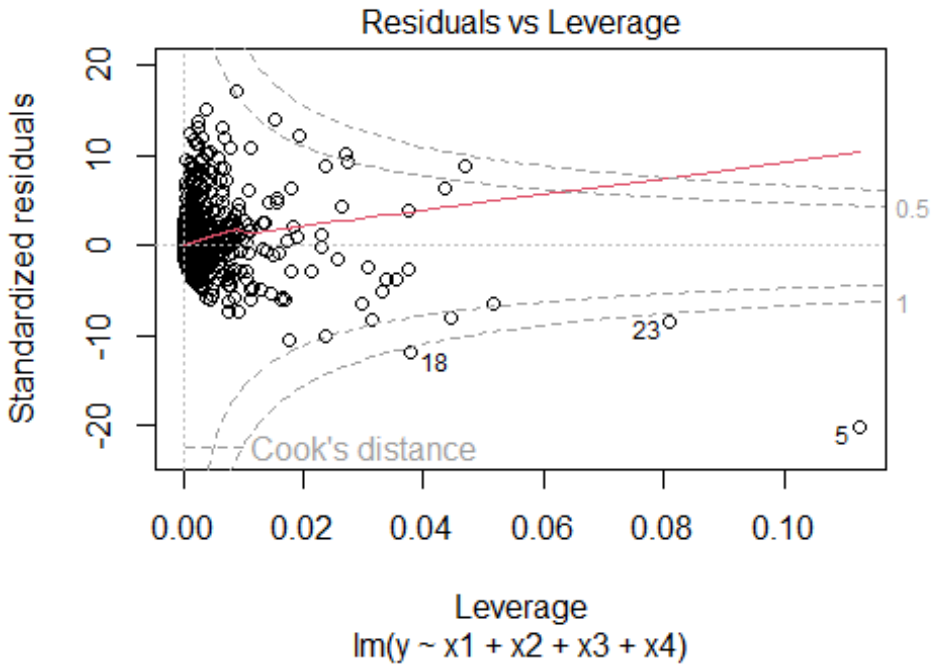
```
## Calls:
## 1: lm(formula = y ~ x1 + x2 + x3 + x4)
## 2: lm(formula = y ~ x1 + x2 + x3)
##
##               Model 1   Model 2
## (Intercept)     0.332    -1.786
## SE              0.112     0.124
##
## x1              1.5484    2.3901
## SE              0.0273    0.0277
##
## x2          -1.17e-06  2.41e-06
## SE           1.46e-07  1.57e-07
##
## x3          -0.065516  0.042001
## SE           0.001837  0.000764
##
## x4           3.24e-07
## SE           5.17e-09
##
```

Коефіцієнти відрізняються.

## Завдання 6: Перевірити дані на наявність аномальних чи високоефективних точок.

**(A)** Представити графічно plot(*, 5);

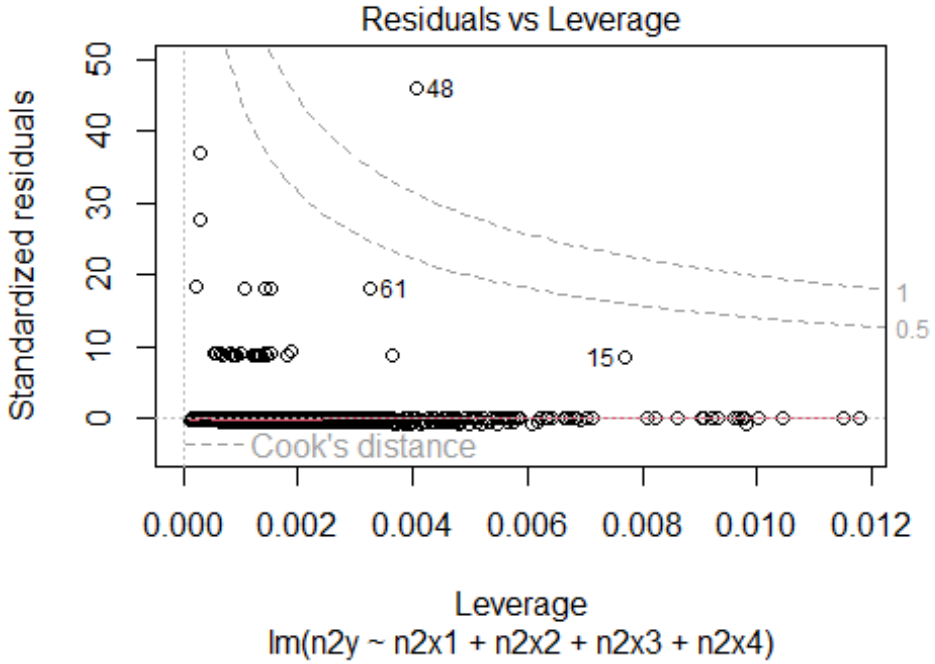
```
plot(mod, 5)
```




**(B)** Видалити аномальну точку та побудувати plot(*, 5);

```
# обчислення медіани та міжквартильного діапазону для всіх змінних
median_x1 <- median(df$Peak_position_times)
iqr_x1 <- IQR(df$Peak_position_times)
lower_x1 <- median_x1 - 1.5 * iqr_x1
upper_x1 <- median_x1 + 1.5 * iqr_x1

median_x2 <- median(df$Peak_streams)
iqr_x2 <- IQR(df$Peak_streams)
lower_x2 <- median_x2 - 1.5 * iqr_x2
upper_x2 <- median_x2 + 1.5 * iqr_x2

median_x3 <- median(df$Days)
iqr_x3 <- IQR(df$Days)
lower_x3 <- median_x3 - 1.5 * iqr_x3
upper_x3 <- median_x3 + 1.5 * iqr_x3

median_x4 <- median(df$Total_streams)
iqr_x4 <- IQR(df$Total_streams)
lower_x4 <- median_x4 - 1.5 * iqr_x4
```

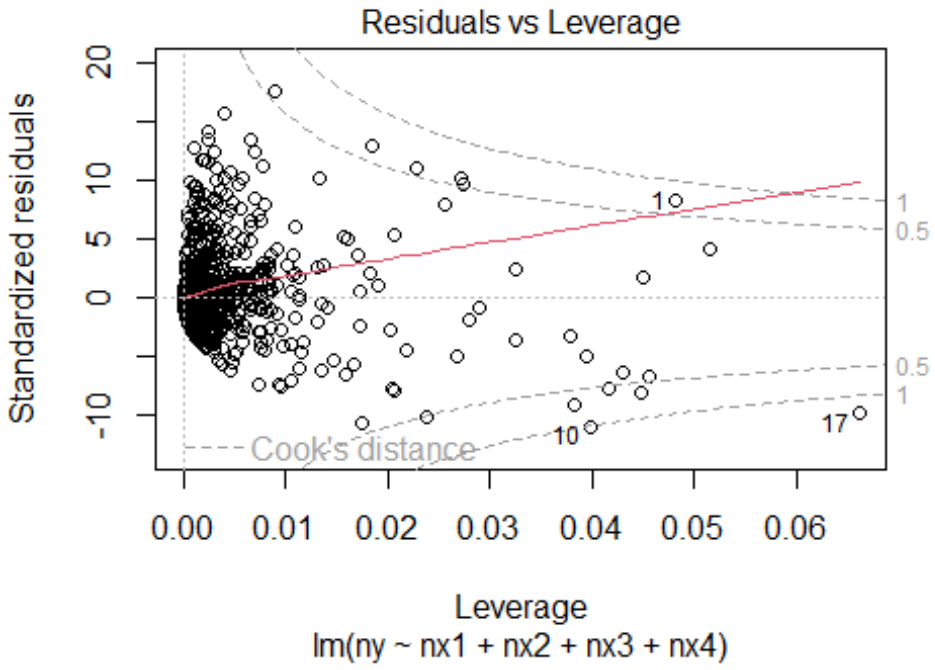
```r
upper_x4 <- median_x4 + 1.5 * iqr_x4


# знаходження аномальних точок для кожної змінної
anomaly_x1 <- which(df$Peak_position_times < lower_x1 | df$Peak_position_times >
upper_x1)
anomaly_x2 <- which(df$Peak_streams < lower_x2 | df$Peak_streams > upper_x2)
anomaly_x3 <- which(df$Days < lower_x3 |  df$Days > upper_x3)
anomaly_x4 <- which(df$Total_streams < lower_x4 | df$Total_streams > upper_x4)

# об'єднання всіх аномальних точок
anomaly <- unique(c(anomaly_x1, anomaly_x2, anomaly_x3, anomaly_x4))

# видалення аномальних точок
dfn2 <- df[-anomaly, ]

n2y <- dfn2$Top_ten_times
n2x1 <- dfn2$Peak_position_times
n2x2 <- dfn2$Peak_streams
n2x3 <- dfn2$Days
n2x4 <- dfn2$Total_streams

mod_normalized <- lm(n2y~n2x1+n2x2+n2x3+n2x4)
plot(mod_normalized, 5)
```




Яка краса, після видалення аномальних точок графік майже ідеальний

**(C)** Видалити високоефективну точку та побудувати plot(*, 5);

```
# обчислення Cook's distance для кожної точки
cooksd <- cooks.distance(mod)
# знайти точки з високими значеннями Cook's distance
high_cooksd <- which(cooksd > 1)
# видалення точок з високими значеннями Cook's distance
dfn3 <- df[-high_cooksd, ]

ny <- dfn3$Top_ten_times
nx1 <- dfn3$Peak_position_times
nx2 <- dfn3$Peak_streams
nx3 <- dfn3$Days
nx4 <- dfn3$Total_streams

mod_normalized <- lm(ny~nx1+nx2+nx3+nx4)
plot(mod_normalized, 5)
```




Після видалення високоефективних точок графік став виглядати краще.