Memoria Econometría:

Rafael Cordoba Martinez
Alejadro de la Hera Luis
David Martinez Diaz
Salvador Molina Plaza

En esta memoria se muestran los resultados de todos los tests vistos en clase para distintas versiones de nuestro modelo inicial. En primer lugar, se muestra nuestro modelo inicial teniendo en cuenta todas las variables.

A continuación, se muestran los resultados para otras dos versiones de nuestro modelo, las cuales nos dimos cuenta que mejoraba los resultados en algunos tests con respecto al modelo inicial.

Por último, se encuentra el modelo final, aquel que conseguía mejores resultados en todos los tests.

Primer Modelo de nuestro proyecto (todas las variables):

Una vez cargado el modelo, vamos a ver los resultados de regresión:

OLS Regression Results

Dep. Variable:	y	R-squared:	0.979
Model:	OLS	Adj. R-squared:	0.975
Method:	Least Squares	F-statistic:	256.6
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	6.59e-35
Time:	12:46:05	Log-Likelihood:	-481.95
No. Observations:	54	AIC:	981.9
Df Residuals:	45	BIC:	999.8
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-120.3937	439.463	-0.274	0.785	-1005.518	764.730
Total Cases	0.3061	0.068	4.499	0.000	0.169	0.443
Death/1 mil population	1.3528	1.280	1.057	0.296	-1.225	3.931
Active Cases	-0.2785	0.072	-3.889	0.000	-0.423	-0.134
Total Cases/1 mil population	-0.0115	0.012	-0.937	0.354	-0.036	0.013
Total Tests	-0.0009	0.000	-1.950	0.057	-0.002	3.03e-05
Tests/1 mil population	-0.0021	0.003	-0.754	0.455	-0.008	0.003
Total Recovered	-0.2794	0.071	-3.924	0.000	-0.423	-0.136
Population	-5.166e-07	9.33e-06	-0.055	0.956	-1.93e-05	1.83e-05

Omnibus:	21.755	Durbin-Watson:	2.074
Prob(Omnibus):	0.000	Jarque-Bera (JB):	197.095
Skew:	-0.226	Prob(JB):	1.59e-43
Kurtosis:	12.348	Cond. No.	7.23e+07

- En este modelo, tenemos un R-squared bastante aceptable con un valor: 0.979.
- También podemos ver que es un modelo significativo o fiable porque su Prob (F-statistic): 6.59e-35, esto quiere decir que cuanto mas bajo sea el valor más fiable.

Test de linealidad:

- Test de Harvey-Collier:

- Estos test nos permite saber si un modelo sigue una disposición lineal, a la hora de diagnosticar los valores de los residuos.

```
from scipy import stats
skip = len(mco1.params)  # bug in linear_harvey_collier
rr = sms.recursive_olsresiduals(mco1, skip=skip, alpha=0.95, order_by=None)
stats.ttest_1samp(rr[3][skip:], 0)
```

✓ 0.9s

```
Ttest_1sampResult(statistic=-0.8166995544901541, pvalue=0.41850016172762494)
```

Como el valor de p es: 0.41850016172762494 al ser mayor > 0.05, aceptamos la hipótesis.

- Test de RESET de Ramsey:

```
print("Test de RESET de Ramsey: ")
rr=oi.reset_ramsey(mco1, degree=2)
rr
```

✓ 0.4s

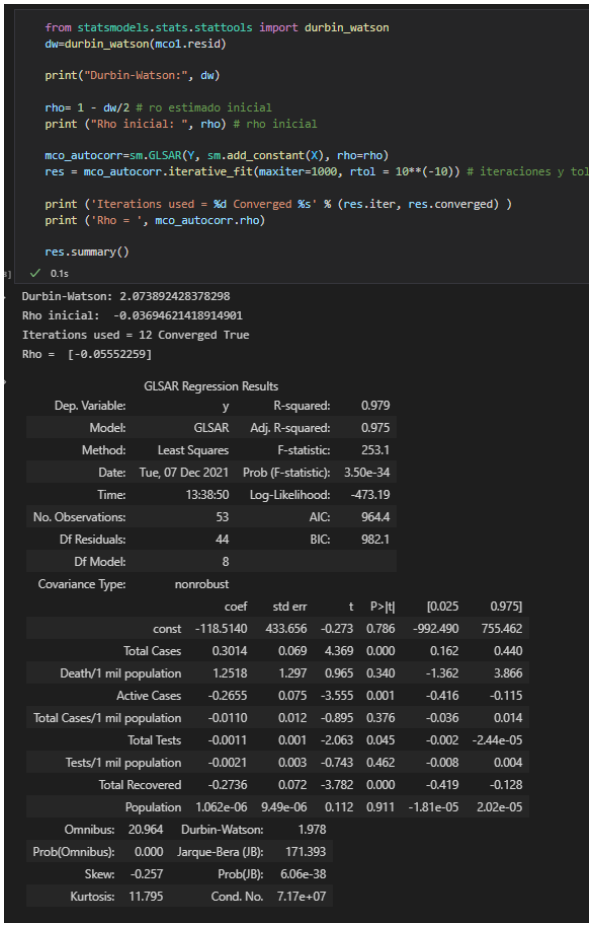
```
Test de RESET de Ramsey:

<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=array([[13.06613404]]), p=0.0007690485400297567, df_denom=44, df_num=1>
```

Como podemos ver que el p-valor en nuestro caso es de p=0.0007690485400297567, podemos decir que no aceptamos la hipótesis porque su valor es menor que 0.05.

Test de Autocorrelación:

- Test de Durbin Watson:

1.- Si el valor es 0, existe una correlación bastante fuerte (positivamente).
2.- Si el valor es 4, existe una correlación bastante fuerte (negativamente).
3.- Si está cerca de 2, quiere decir que no existe correlación.
4.- Miramos en la tabla de DW para comprobar su valor.



```
from statsmodels.stats.stattools import durbin_watson
dw=durbin_watson(mco1.resid)

print("Durbin-Watson:", dw)

rho= 1 - dw/2 # ro estimado inicial
print ("Rho inicial: ", rho) # rho inicial

mco_autocorr=sm.GLSAR(Y, sm.add_constant(X), rho=rho)
res = mco_autocorr.iterative_fit(maxiter=1000, rtol = 10**(-10)) # iteraciones y tol

print ('Iterations used = %d Converged %s' % (res.iter, res.converged) )
print ('Rho = ', mco_autocorr.rho)

res.summary()
```

✓ 0.1s

```
Durbin-Watson: 2.073892428378298
Rho inicial:  -0.03694621418914901
Iterations used = 12 Converged True
Rho =  [-0.05552259]
```

GLSAR Regression Results

Dep. Variable:	y	R-squared:	0.979
Model:	GLSAR	Adj. R-squared:	0.975
Method:	Least Squares	F-statistic:	253.1
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	3.50e-34
Time:	13:38:50	Log-Likelihood:	-473.19
No. Observations:	53	AIC:	964.4
Df Residuals:	44	BIC:	982.1
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-118.5140	433.656	-0.273	0.786	-992.490	755.462
Total Cases	0.3014	0.069	4.369	0.000	0.162	0.440
Death/1 mil population	1.2518	1.297	0.965	0.340	-1.362	3.866
Active Cases	-0.2655	0.075	-3.555	0.001	-0.416	-0.115
Total Cases/1 mil population	-0.0110	0.012	-0.895	0.376	-0.036	0.014
Total Tests	-0.0011	0.001	-2.063	0.045	-0.002	-2.44e-05
Tests/1 mil population	-0.0021	0.003	-0.743	0.462	-0.008	0.004
Total Recovered	-0.2736	0.072	-3.782	0.000	-0.419	-0.128
Population	1.062e-06	9.49e-06	0.112	0.911	-1.81e-05	2.02e-05

Omnibus:	20.964	Durbin-Watson:	1.978
Prob(Omnibus):	0.000	Jarque-Bera (JB):	171.393
Skew:	-0.257	Prob(JB):	6.06e-38
Kurtosis:	11.795	Cond. No.	7.17e+07

Al ser un valor próximo al 2, podemos aceptar este modelo. Nuestros residuos están incorrelados.

Podemos comentar que para aceptar este modelo la aproximación que se realiza entre el Rho inicial y el Rho final debe ser pequeña.

- Test de Ljung-box:

1.- Con este test podemos probar si existe relación en mi perturbación aleatoria con la de x años antes.
2.- Los lags son los retardos, pudiendo poner cualquier número de años.
3.- Explicación: da el p valor, que es la siguiente lista, acepto las 3 hipótesis ya que todos los p valores son más grande que la alfa (0.05).
4.- No deberíamos tener problema de autocorrelación.

```
from statsmodels.stats.diagnostic import acorr_ljungbox # analizamos con Ljung-Box si
mco2=sm.OLS(Y, sm.add_constant(X)).fit()
acorr_ljungbox(mco2.resid, lags=3, return_df=True) # lags son los retardos, comparo h
```

✓ 0.4s

	lb_stat	lb_pvalue
1	0.109440	0.740783
2	0.126071	0.938910
3	0.640474	0.887107

Como los valores del primer array con respectos a los del segundo array, en nuestro caso, como son valores no extremadamente chicos podemos aceptar la hipótesis.

Test de Heterocedasticidad:

En un modelo hay heterocedasticidad cuando la varianza de los errores no es igual en todas las observaciones realizadas.

Por lo que buscamos comprobar así, si se cumple uno de los requisitos básicos de las hipótesis de los modelos lineales.

- Test de goldfelquandt (test para muestra pequeñas):

- Split es el número de elementos que pone en cada trozo, si no pones nada divide en 2 por defecto.
- Para dividir entre 3 las variables, para comparar los maximos y minimos.

```
import statsmodels.stats.api as sms

#Test goldfelquandt , split es el numero de elementos que pone en cada trozo, si no pone
#Para dividir entre 3 las variables, para comparar los maximos y minimos, tampoco tengo

GQ = sms.het_goldfeldquandt(mco1.resid, mco1.model.exog)

#Hipotesis nula homocedasticidad, no tengo un problema de heterocedasticidad
#Todo numero mayor que 0.05 rechazo entonces tengo heterocedasticidad

print (GQ)
```

✓ 0.3s

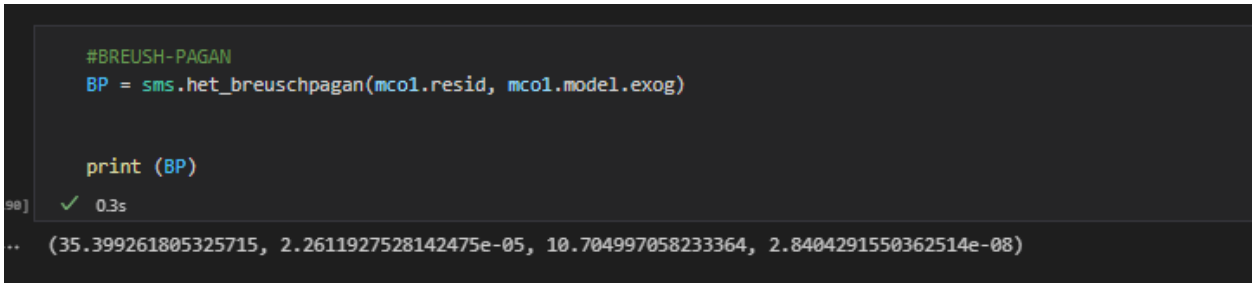
```
(1.2147070936011652, 0.337956702000184, 'increasing')
```

Este test nos sirve para modelos con pocos datos, como es nuestro caso, podemos observar que nuestro p-valor es aceptable ya que supera el valor de 0.05, por lo que aceptamos la hipótesis nula de homocedasticidad.

p-valor: 0.337956702000184 > 0.05.

- Test de Breush-Pagan (test para muestras grandes):

- Los dos últimos valores me interesan , el estadístico experimental y P valor:
  - (como este es mayor que 0.05 no tengo problema de heterocedasticidad) tengo homocedasticidad porque nuestro valor es más pequeño 0,05.
  - (alpha nivel de significación a partir del cual empiezo a rechazar el modelo).

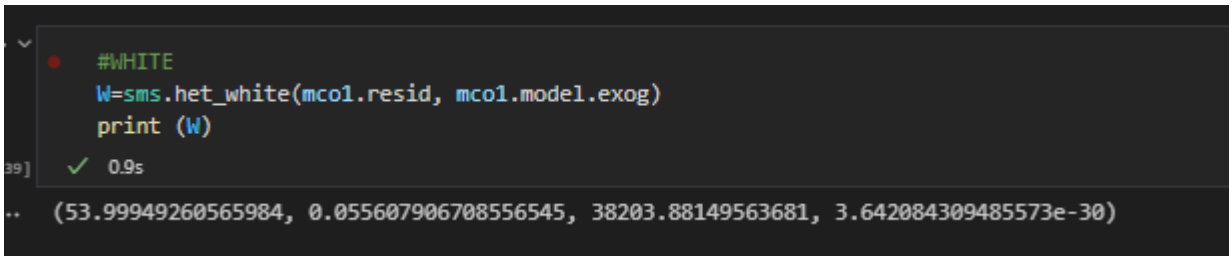


Estos test son para modelos con un número significativo de muestras, aun así nuestro p-valor sigue siendo muy bajo por lo que tenemos problemas de heterocedasticidad, al rechazar la hipótesis nula.

p-valor: 2.8404291550362514e-08 < 0.05

- Test de White:

- El test de white es muy parecido, pero tiene en cuenta el producto de las variables.
- Si me da significación global no tengo problema de heterocedasticidad.



En este caso, pasa lo mismo, nuestro p-valor es menor que alfa, por lo que tendríamos problemas de heterocedasticidad:

p-valor: 3.642084309485573e-30 < 0.05.

- Test de Glejser:

- Vamos a comprobar todas estas variables y nos vamos a quedar con la que tenga el R2 más grande, si este valor es mayor que 0.05.
- Ese será nuestro Z^2, el cual se puede corregir con mínimos cuadrados ponderados, multiplicando mis datos por 1/sqrt(Z^2).

```
maximo=""
alfa=0.05
r_maximo=0.0

#
xnames=[ "Total Cases", "Death/1 mil population", "Total Recovered", "Active Cases", "Total

for i in xnames:
    print("-------------------------")
    print("Para la variable: " + i)
    z=data[i]
    z = z.astype('float64')
    for h in [-2,-1,-0.5,0.5,1,2]:
        mco_glejser=sm.OLS(abs(mco1.resid),sm.add_constant((z**h))).fit()
        p_valor=mco_glejser.pvalues[1]
        r2=mco_glejser.rsquared
        print("h: ",h,"-> pvalt:",p_valor,"R2: ", mco_glejser.rsquared)
        if((p_valor<alfa) and (r2>r_maximo)):
            r_maximo=r2
            maximo=i
    print("-------------------------")

print(maximo + ": provoca problema en la heterocedasticidad.")
mcp = sm.WLS(Y, sm.add_constant(X),weights=1./np.sqrt (z**2)).fit()
mcp.summary()
```

✓ 0.3s

```
-------------------------
Para la variable: Total Cases
h:  -2 -> pvalt: 0.12421367572900914 R2:  0.04484991589248377
h:  -1 -> pvalt: 0.0363826880880016695 R2:  0.08150680346572259
h:  -0.5 -> pvalt: 0.0103769838647469O4 R2:  0.11973683853778339
h:  0.5 -> pvalt: 0.0006915159807027S5 R2:  0.13588905079103264
h:  1 -> pvalt: 0.07699910375881404 R2:  0.058908739382352415
h:  2 -> pvalt: 0.4372504275341985 R2:  0.011649159916995888
-------------------------
-------------------------
Para la variable: Death/1 mil population
h:  -2 -> pvalt: 0.5164750991907099 R2:  0.008139583736734513
h:  -1 -> pvalt: 0.39566812207128943 R2:  0.013910298162193135
h:  -0.5 -> pvalt: 0.3079007098639721 R2:  0.019983900629654938
h:  0.5 -> pvalt: 0.46729848226789225 R2:  0.010206346448498604
h:  1 -> pvalt: 0.7305085247069362 R2:  0.002301026465478806
h:  2 -> pvalt: 0.9335096824360574 R2:  0.0001351408917700247S
-------------------------
-------------------------
Para la variable: Total Recovered
h:  -2 -> pvalt: 2.708912377246719e-05 R2:  0.2895577908114948
h:  -1 -> pvalt: 2.761571234904313e-05 R2:  0.28905284704054246
h:  -0.5 -> pvalt: 5.922956613737545e-05 R2:  0.26878968545938164
h:  0.5 -> pvalt: 0.022044263682366432 R2:  0.09677069972015839
h:  1 -> pvalt: 0.09925140790276134 R2:  0.05139506376347802
```

```
h:  0.5 -> pvalt: 0.0039027467123639572 R2:  0.1492951623403913
h:  1 -> pvalt: 0.005676316059147533 R2:  0.13802111614657986
h:  2 -> pvalt: 0.009355853945953976 R2:  -0.18507264437581172
-------------------------
Total Tests: provoca problema en la heterocedasticidad.
```

WLS Regression Results

Dep. Variable:	y	R-squared:	0.946
Model:	WLS	Adj. R-squared:	0.940
Method:	Least Squares	F-statistic:	167.4
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	3.72e-29
Time:	18:04:17	Log-Likelihood:	-468.53
No. Observations:	54	AIC:	949.1
Df Residuals:	48	BIC:	961.0
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	94.2346	194.957	0.483	0.631	-297.752	486.222
Total Recovered	0.0295	0.001	22.708	0.000	0.027	0.032
Death/1 mil population	0.9318	0.536	1.737	0.089	-0.147	2.010
Total Cases/1 mil population	-0.0049	0.003	-1.941	0.058	-0.010	0.000
Tests/1 mil population	-0.0030	0.001	-3.618	0.001	-0.005	-0.001
Population	-7.075e-06	1.56e-05	-0.452	0.653	-3.85e-05	2.44e-05

Omnibus:	25.144	Durbin-Watson:	2.092
Prob(Omnibus):	0.000	Jarque-Bera (JB):	89.043
Skew:	-1.075	Prob(JB):	4.62e-20
Kurtosis:	8.912	Cond. No.	1.52e+07

Como vemos en la imagen, el test de Glejser nos dice que la variable que genera más problemas de heterocedasticidad es Total Tests, y nos da un summary de cómo sería nuestro modelo sin esta variable.

Test de Multicolinealidad:

- Condition number:

```
print("Condition number: ", mco1.condition_number) #Número de Condición

#Tendriamos problemas de multicolinealidad si fuese mayor que 900, ya que esta al cuadrado
# Tenemos un problema gordo porque nuestro valor > 900.
```

✓ 0.3s

```
Condition number:  72311132.56520036
```

Nuestro condition number es altísimo, lo que quiere decir que nuestro modelo presenta variables que dependen unas de otras.

Condition number: 72311132.56520036 > 900

Como nuestro condition number es mayor que 900 tenemos una dependencia entre variables y un problema grave.

- Test de Vifs:

- Comprobamos el factor de inflación de la varianza en todas las variables.
    - Posibles resultados:
        - Si sale mayor que 10: tenemos problema de multicolinealidad.
        - Si sale mayor que 20: problema gordo de multicolinealidad.

```
import statsmodels.stats.outliers_influence as oi

vifs=[oi.variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

vifs
```

✓ 0.4s

```
[12248.616231591255,
 5.340629511164673,
 20.33889974641905,
 2.7210239838532577,
 28.31830563539725,
 3.043256827332228,
 12222.703904925464,
 1.9253733559700568]
```

En este caso, para ciertas variables nos tenemos un problema gordo, ya que nos salen unos valores enormes y presentamos problemas de correlación en 4 variables.

Es decir, son las variables que dependen unas de otras.

- Matriz de Correlaciones:

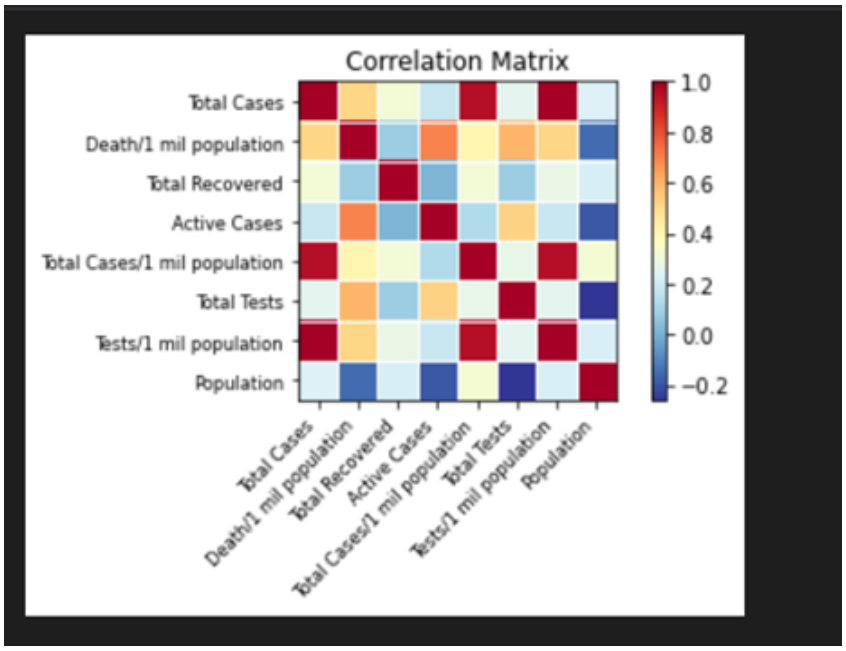
- Se busca que la matriz se parezca a la matriz identidad para sacar como factor común la varianza como la multiplicación de esta por la identidad.

```
#Matriz de correlacciones
corr_matrix=np.corrcoef(X.T) #X.T = matriz X traspuesta
print(corr_matrix)
```

✓ 0.3s

```
[[ 1.          0.51954038  0.31659959  0.18571537  0.95778797  0.26299201
   0.99933211  0.23753991]
 [ 0.51954038  1.          0.08539872  0.70020222  0.40493669  0.59659099
   0.52177148 -0.15004533]
 [ 0.31659959  0.08539872  1.          0.01973277  0.32002308  0.08419675
   0.28327872  0.22591983]
 [ 0.18571537  0.70020222  0.01973277  1.          0.1293295   0.52916249
   0.18781035 -0.19110093]
 [ 0.95778797  0.40493669  0.32002308  0.1293295   1.          0.27776218
   0.95771343  0.32629645]
 [ 0.26299201  0.59659099  0.08419675  0.52916249  0.27776218  1.
   0.26473752 -0.26075754]
 [ 0.99933211  0.52177148  0.28327872  0.18781035  0.95771343  0.26473752
   1.          0.23077628]
 [ 0.23753991 -0.15004533  0.22591983 -0.19110093  0.32629645 -0.26075754
   0.23077628  1.        ]]
```

Nuestra matriz se asemeja a la matriz de identidad, en cierto sentido, por lo que podemos decir que si se cumple lo que sea que se tiene que cumplir.



Con este gráfico podemos ver las variables que muestran los mismos datos o que no son tan necesarios ya que la información que aportan es similar a la que muestra su relación.

- Normalidad de los Residuos:

- Jarque-Bera: Test de hipótesis que contrasta si los datos de la muestra tienen el coeficiente de simetría y la curtósis de una distribución normal.
- chi^2 (p-valor): p-valor del Test de Jaque-Bera.
- Skew: Coeficiente de Simetría de pearson de los residuos.
- Kurtosis: Coeficiente de apuntamiento de los residuos.

    - Análisis de los resultados:

        - Para que aceptemos la hipótesis de la normalidad nula (Chi-cuadrado y Jarque-Bera), nuestro p-valor debe ser mayor que alfa (0.05).
        - Para que aceptemos la hipótesis de la simetría de skew, nuestro valor tiene que ser próximo a 0.
        - Para que aceptemos la hipótesis de kurtosis, el valor debe ser próximo a 4.

```
import statsmodels.stats.api as sms
mco_res = sm.OLS(Y, sm.add_constant(X)).fit()
name = ['Jarque-Bera', 'Chi^2 two-tail prob.', 'Skew', 'Kurtosis']
test = sms.jarque_bera(mco_res.resid)
for i in range(4):
    print(name[i], test[i])
```

✓ 0.6s

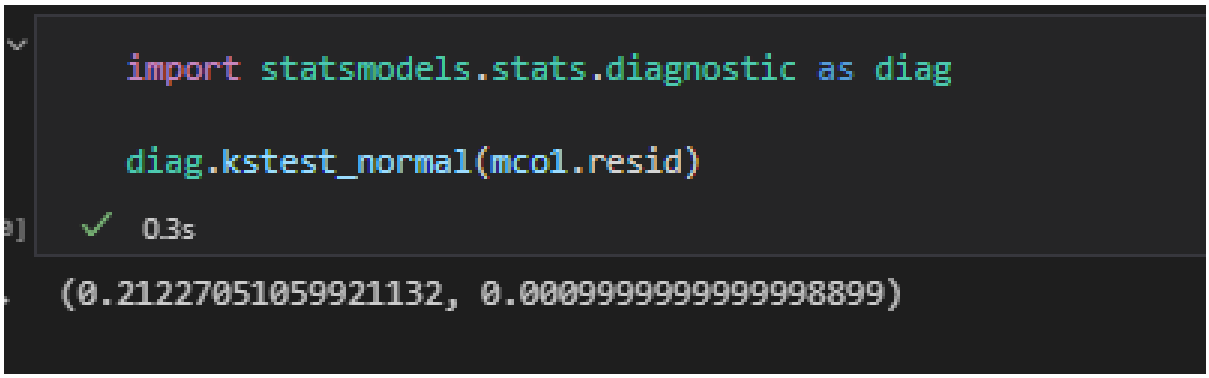
```
Jarque-Bera 197.0946281344043
Chi^2 two-tail prob. 1.5901763651229992e-43
Skew -0.22601131044766026
Kurtosis 12.348437733353023
```

- Como nuestro Chi-cuadrado es: 5.192827366266701e-71 < 0.05, no podemos aceptar la hipótesis de la normalidad nula.
- El valor de nuestro Skew es: 1.5756829717965635, no podemos decir que es sigue un modelo simétrico ya que es no es próximo a 0.
- Nuestro valor de kurtosis es: 14.572525607665003, por lo que deberías rechazarlo ya que no es próximo a 4.

- Graficas:

  - Como nuestros datos al menos la mitad de ellos se encuentran por encima de la media, no están del todo mal.

Kolmogorov-Smirnov: Test de Hipótesis para contrastar si una muestra proviene de una distribución (en este caso normal):



Como nuestro p-valor: 0.0009999999999998899 < 0.05, rechazamos la hipótesis.

Segundo Modelo de nuestro proyecto (eliminando la variable Total Cases)

Primero podemos observar el summary:

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.969
Model:	OLS	**Adj. R-squared:**	0.964
Method:	Least Squares	**F-statistic:**	204.7
Date:	Tue, 07 Dec 2021	**Prob (F-statistic):**	1.83e-32
Time:	13:18:51	**Log-Likelihood:**	-491.97
No. Observations:	54	**AIC:**	999.9
Df Residuals:	46	**BIC:**	1016.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	45.1709	521.526	0.087	0.931	-1004.606	1094.947
Death/1 mil population	1.5659	1.523	1.028	0.309	-1.500	4.632
Active Cases	0.0319	0.023	1.392	0.171	-0.014	0.078
Total Cases/1 mil population	-0.0160	0.015	-1.103	0.276	-0.045	0.013
Total Tests	-0.0017	0.001	-3.145	0.003	-0.003	-0.001
Tests/1 mil population	-0.0025	0.003	-0.760	0.451	-0.009	0.004
Total Recovered	0.0406	0.004	10.696	0.000	0.033	0.048
Population	8.521e-06	1.08e-05	0.786	0.436	-1.33e-05	3.04e-05

Omnibus:	40.708	**Durbin-Watson:**	2.050
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	323.673
Skew:	1.576	**Prob(JB):**	5.19e-71
Kurtosis:	14.573	**Cond. No.**	7.21e+07

- En este modelo, tenemos un R-squared bastante aceptable con un valor: 0.969.

- También podemos ver que es un modelo significativo o fiable porque su Prob (F-statistic): 1.83e-32, esto quiere decir que cuanto mas bajo sea el valor más fiable.

Test de linealidad:

- Test de Harvey-Collier:

- Estos test nos permite saber si un modelo sigue una disposición lineal, a la hora de diagnosticar los valores de los residuos.

```
print("Test de Harvey Collier: ")
from scipy import stats
skip = len(mco1.params)  # bug in linear_harvey_collier
rr = sms.recursive_olsresiduals(mco1, skip=skip, alpha=0.95, order_by=None)
stats.ttest_1samp(rr[3][skip:], 0)
```

```
Test de Harvey Collier:

Ttest_1sampResult(statistic=0.5432792927907356, pvalue=0.5896187631447047)
```

Como el valor de p es: 0.0.5896187631447047 al ser mayor > 0.05, aceptamos la hipotesis.

- Test de RESET de Ramsey:

```
print("Test de RESET de Ramsey: ")
rr=oi.reset_ramsey(mco1, degree=2)
rr
```

```
Test de RESET de Ramsey:

<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=array([[18.88356265]]), p=7.830122526345829e-05, df_denom=45, df_num
=1>
```

Como el valor de p es extremedamente pequeño no podemos aceptar la hipotesis ya que esta deberia ser mayor que 0.05

Test de Autocorrelación:

- Test de Durbin Watson:

1.- Si el valor es 0, existe una correlación bastante fuerte (positivamente).
2.- Si el valor es 4, existe una correlación bastante fuerte (negativamente).
3.- Si está cerca de 2, quiere decir que no existe correlación.
4.- Miramos en la tabla de DW para comprobar su valor.

```
from statsmodels.stats.stattools import durbin_watson
dw=durbin_watson(mco1.resid)

print("Durbin-Watson:", dw)

rho= 1 - dw/2 # ro estimado inicial
print ("Rho inicial: ", rho) # rho inicial

mco_autocorr=sm.GLSAR(Y, sm.add_constant(X), rho=rho)
res = mco_autocorr.iterative_fit(maxiter=1000, rtol = 10**(-10)) # iteraciones

print ('Iterations used = %d Converged %s' % (res.iter, res.converged) )
print ('Rho = ', mco_autocorr.rho)

res.summary()
```

```
Durbin-Watson: 2.049775773541431
Rho inicial:  -0.024887886770715406
Iterations used = 19 Converged True
Rho =  [-0.06048542]
```

GLSAR Regression Results

Dep. Variable:	y	R-squared:	0.970
Model:	GLSAR	Adj. R-squared:	0.965
Method:	Least Squares	F-statistic:	204.4
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	6.34e-32
Time:	13:32:15	Log-Likelihood:	-482.73
No. Observations:	53	AIC:	981.5
Df Residuals:	45	BIC:	997.2
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	68.9975	510.058	0.135	0.893	-960.312	1094.307
Death/1 mil population	1.3340	1.536	0.869	0.390	-1.759	4.427
Active Cases	0.0451	0.027	1.670	0.102	-0.009	0.100
Total Cases/1 mil population	-0.0151	0.015	-1.039	0.304	-0.044	0.014
Total Tests	-0.0019	0.001	-3.387	0.002	-0.003	-0.001
Tests/1 mil population	-0.0025	0.003	-0.747	0.459	-0.009	0.004
Total Recovered	0.0421	0.004	10.558	0.000	0.034	0.050
Population	1.115e-05	1.09e-05	1.022	0.312	-1.08e-05	3.31e-05

Omnibus:	38.417	Durbin-Watson:	1.973
Prob(Omnibus):	0.000	Jarque-Bera (JB):	287.420
Skew:	1.379	Prob(JB):	8.52e-63
Kurtosis:	13.653	Cond. No.	7.13e+07

Al ser un valor próximo al 2, podemos aceptar que este modelo. Nuestros residuos están incorrelados. DU < DW < 4 - DU

Podemos comentar que para aceptar este modelo la aproximación que se realiza entre el Rho inicial y el Rho final debe ser pequeña.

- Test de Ljung-Box:

1.- Con este test podemos probar si existe relación en mi perturbación aleatoria con la de x años antes.
2.- Los lags son los retardos, pudiendo poner cualquier número de años.
3.- Explicación: da el p valor, que es la siguiente lista, rechazo las hipótesis si los p-valores son extremadamente chicos, es decir existe relación con los otros años.

```
: from statsmodels.stats.diagnostic import acorr_ljungbox # analizamos con Lj
mco2=sm.OLS(Y, sm.add_constant(X)).fit()
acorr_ljungbox(mco2.resid, lags=3) # Lags son los retardos, comparo hasta c

#
```

```
C:\Users\Alejandro de la Hera\anaconda3\lib\site-packages\statsmodels\stats\
gnostic.py:559: FutureWarning: The value returned will change to a single Da
rame after 0.12 is released.  Set return_df to True to use to return a Datai
e now.  Set return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
```

```
: (array([0.07625965, 0.12891632, 0.23083807]),
   array([0.78243171, 0.93757534, 0.97246383]))
```

Como los valores del primer array con respectos a los del segundo array, en nuestro caso, como son valores no extremadamente chicos podemos aceptar la hipótesis.

Test de Heterocedasticidad:

En un modelo hay heterocedasticidad cuando la varianza de los errores no es igual en todas las observaciones realizadas.

Por lo que buscamos comprobar así, si se cumple uno de los requisitos básicos de las hipótesis de los modelos lineales.

- Test de goldfelquandt (test para muestra pequeñas):

- Split es el número de elementos que pone en cada trozo, si no pones nada divide en 2 por defecto.
- Para dividir entre 3 las variables, para comparar los maximos y minimos.

```
import statsmodels.stats.api as sms

#Test goldfelquandt , split es el numero de elementos que pone en cada trozo
#Para dividir entre 3 las variables, para comparar los maximos y minimos, ta

GQ = sms.het_goldfeldquandt(mco1.resid, mco1.model.exog)

#Hipotesis nula homocedasticidad, no tengo un problema de heterocedasticidad
#Todo numero mayor que 0.05 rechazo entonces tengo heterocedasticidad

print (GQ)
```

```
(1.2147070936011652, 0.337956702000184, 'increasing')
```

Este test nos sirve para modelos con pocos datos, como es nuestro caso, podemos observar que nuestro p-valor es aceptable ya que supera el valor de 0.05, por lo que aceptamos la hipótesis nula de homocedasticidad.

p-valor: 0.337956702000184 > 0.05

- ## Test de Breush-Pagan (test para muestras grandes):

  - Los dos últimos valores me interesan , el estadístico experimental y P valor:
    - (como este es mayor que 0.05 no tengo problema de heterocedasticidad) tengo homocedasticidad porque nuestro valor es más pequeño 0,05.
    - (alpha nivel de significación a partir del cual empiezo a rechazar el modelo).

Test para muestras mas grandes

```
#BREUSH-PAGAN
BP = sms.het_breuschpagan(mco1.resid, mco1.model.exog)

#Los dos ultimos valores me interesan , el estadistico experimental y P valo
# (como este es mayor que 0.05 no tengo probelma de heterocedasticidad) teng
# (alpha nivel de significacion a partir del cual empiezo a rechazar el mode
print (BP)
```

```
(18.964649605089956, 0.008298803788939586, 3.5571169934731204, 0.0038800935
85801874)
```

Estos test son para modelos con un número significativo de muestras, aun así nuestro p-valor sigue siendo muy bajo por lo que tenemos problemas de heterocedasticidad, al rechazar la hipótesis nula.

p-valor: 0.0038800935858018 74 < 0.05

- ## Test de White:

  - El test de white es muy parecido, pero tiene en cuenta el producto de las variables.
  - Si me da significación global no tengo problema de heterocedasticidad.

Test para muestras mas grandes

```
]: #WHITE

#El test de white es muy parecido, pero tiene en cuenta el producto de las v
#Si me da significacion global no tengo problema de heterocedasticidad
W=sms.het_white(mco1.resid, mco1.model.exog)

# El P valor 0.19, a partir de eso es cuando rechazo, como mi alpha es 0.05,
# problema de heterocedasticidad, no rechazo la hipotesis nula.

print (W)
```

```
(53.133722289908654, 0.010870471260997796, 40.25153232798243, 5.79746602932
0036e-13)
```

En este caso, pasa lo mismo, como es para muestras grandes no nos sirve y no da un valor no aceptable:

p-valor: 5.797466029320036e-13 < 0.05.

- Test de Glejser:

- Vamos a comprobar todas estas variables y nos vamos a quedar con la que tenga el R2 más grande, si este valor es mayor que 0.05.
- Ese será nuestro Z^2, el cual se puede corregir con mínimos cuadrados ponderados, multiplicando mis datos por 1/sqrt(Z^2).

```
maximo=""
alfa=0.05
r_maximo=0.0

# "Total Cases",
xnames=[  "Death/1 mil population", "Total Recovered", "Active Cases", "Total Cases/1 mil p

for i in xnames:
    print("-------------------------")
    print("Para la variable: " + i)
    z=data[i]
    z = z.astype('float64')
    for h in [-2,-1,-0.5,0.5,1,2]:
        mco_glejser=sm.OLS(abs(mco1.resid),sm.add_constant((z**h))).fit()
        p_valor=mco_glejser.pvalues[1]
        r2=mco_glejser.rsquared
        print("h: ",h,"-> pvalt:",p_valor,"R2: ", mco_glejser.rsquared)
        if((p_valor<alfa) and (r2>r_maximo)):
            r_maximo=r2
            maximo=i
    print("-------------------------")

print(maximo + ": provoca problema en la heterocedasticidad.")
mcp = sm.WLS(Y, sm.add_constant(X),weights=1./np.sqrt (z**2)).fit()
mcp.summary()
```

✓ 0.3s

```
-------------------------
Para la variable: Death/1 mil population
h:  -2 -> pvalt: 0.380893687888073 R2:  0.014797580594749338
h:  -1 -> pvalt: 0.12372456321501422 R2:  0.044964098529434504
h:  -0.5 -> pvalt: 0.05564199757856679 R2:  0.06864406334492223
h:  0.5 -> pvalt: 0.21926628810927995 R2:  0.028876987277321O2
h:  1 -> pvalt: 0.5218713566631048 R2:  0.007932861351753306
h:  2 -> pvalt: 0.9505468563581654 R2:  7.468277423094971e-05
-------------------------
-------------------------
Para la variable: Total Recovered
h:  -2 -> pvalt: 0.6081860764224651 R2:  0.005090134101864319
h:  -1 -> pvalt: 0.6051518720480003 R2:  0.005176490186236227
h:  -0.5 -> pvalt: 0.49242347555098A1 R2:  0.009107800117772125
h:  0.5 -> pvalt: 0.00022815200473263458 R2:  0.2317910481620319
h:  1 -> pvalt: 0.024679248898481936 R2:  0.09332672543977116
h:  2 -> pvalt: 0.302976569117947A6 R2:  0.020390773462880984
-------------------------
-------------------------
Para la variable: Active Cases
h:  -2 -> pvalt: 0.5737342406882536 R2:  0.006125970082663912
h:  -1 -> pvalt: 0.3995343112786902 R2:  0.013685798469844745
h:  -0.5 -> pvalt: 0.10503960513965566 R2:  0.04973197241345495
h:  0.5 -> pvalt: 5.501603631039032e-06 R2:  0.3302856358069429
```

```
h:  0.5 -> pvalt: 0.0009535210980838765 R2:  0.19096189874140468
h:  1 -> pvalt: 0.00121410168719192 R2:  0.18391268705615382
h:  2 -> pvalt: 0.0031406855178418005 R2:  -0.1444989669677954
-------------------------
Active Cases: provoca problema en la heterocedasticidad.
```

WLS Regression Results

Dep. Variable:	y	R-squared:	0.953
Model:	WLS	Adj. R-squared:	0.946
Method:	Least Squares	F-statistic:	134.2
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	2.00e-28
Time:	13:51:39	Log-Likelihood:	-464.47
No. Observations:	54	AIC:	944.9
Df Residuals:	46	BIC:	960.9
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	133.2869	187.032	0.713	0.480	-243.189	509.762
Death/1 mil population	0.2356	0.569	0.414	0.681	-0.909	1.381
Active Cases	-0.0021	0.014	-0.147	0.884	-0.031	0.027
Total Cases/1 mil population	-0.0035	0.002	-1.440	0.157	-0.009	0.001
Total Tests	-0.0011	0.000	-2.665	0.011	-0.002	-0.000
Tests/1 mil population	-0.0014	0.001	-1.490	0.143	-0.003	0.001
Total Recovered	0.0370	0.003	12.154	0.000	0.031	0.043
Population	9.407e-06	1.6e-05	0.587	0.560	-2.28e-05	4.17e-05

Omnibus:	8.211	Durbin-Watson:	2.164
Prob(Omnibus):	0.016	Jarque-Bera (JB):	15.207
Skew:	-0.236	Prob(JB):	0.000499
Kurtosis:	5.556	Cond. No.	1.54e+07

Test de Multicolinealidad:

- Condition number:

```
print("Condition number: ", mco1.condition_number) #Número de Condición

#Tendriamos problemas de multicolinealidad si fuese mayor que 900, ya que
# Tenemos un problema gordo porque nuestro valor > 900.
```

```
Condition number:  72056627.85705274
```

Nuestro condition number es altísimo, lo que quiere decir que nuestro modelo presenta variables que dependen unas de otras.

Condition number: 72056627.85705274 > 900

Como nuestro condition number es mayor que 900 tenemos una dependencia entre variables y un problema grave.

- Test de Vifs:

- Comprobamos el factor de inflación de la varianza en todas las variables.
    - Posibles resultados:
        - Si sale mayor que 10: tenemos problema de multicolinealidad.
        - Si sale mayor que 20: problema gordo de multicolinealidad.

```
import statsmodels.stats.outliers_influence as oi

vifs=[oi.variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

vifs
```

```
[5.325644278695361,
 1.422969408012339,
 2.7018115937264926,
 25.042179839707888,
 3.0430246939203984,
 24.071070191698745,
 1.7760726395962798]
```

En este test, por lo tanto, se ve que hemos conseguido disminuir la dependencia entre las variables, pues ahora, solo hay problemas de correlación con 2 variables.

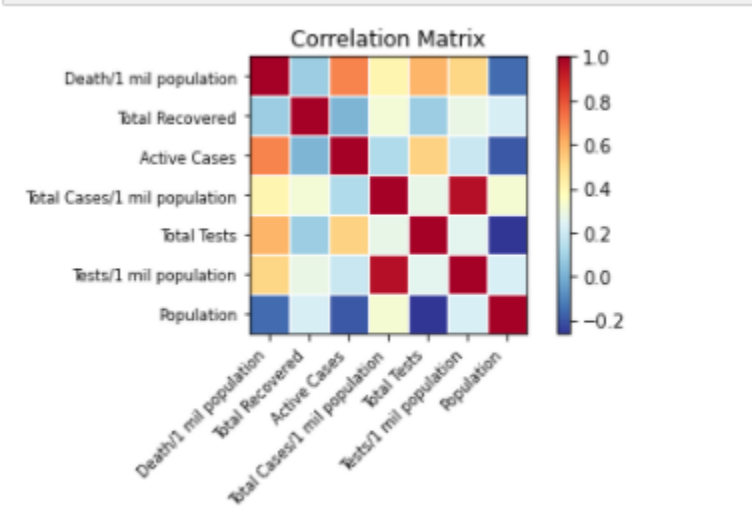
- Matriz de Correlaciones:

- Se busca que la matriz se parezca a la matriz identidad para sacar como factor común la varianza como la multiplicación de esta por la identidad.

```
#Matriz de correlacciones
corr_matrix=np.corrcoef(X.T) #X.T = matriz X traspuesta
print(corr_matrix)
```

```
[[ 1.          0.08539872  0.70020222  0.40493669  0.59659099  0.52177148
  -0.15004533]
 [ 0.08539872  1.          0.01973277  0.32002308  0.08419675  0.28327872
   0.22591983]
 [ 0.70020222  0.01973277  1.          0.1293295   0.52916249  0.18781035
  -0.19110093]
 [ 0.40493669  0.32002308  0.1293295   1.          0.27776218  0.95771343
   0.32629645]
 [ 0.59659099  0.08419675  0.52916249  0.27776218  1.          0.26473752
  -0.26075754]
 [ 0.52177148  0.28327872  0.18781035  0.95771343  0.26473752  1.
   0.23077628]
 [-0.15004533  0.22591983 -0.19110093  0.32629645 -0.26075754  0.23077628
   1.        ]]
```

Nuestra matriz se asemeja a la matriz de identidad, en cierto sentido, por lo que podemos decir que si se cumple lo que sea que se tiene que cumplir.



Se ve reflejado en el gráfico la mejora de la que hablábamos antes, pues ahora las variables no dependen tanto entre sí.

- Normalidad de los Residuos:

- Jarque-Bera: Test de hipótesis que contrasta si los datos de la muestra tienen el coeficiente de simetría y la curtósis de una distribución normal.
- chi^2 (p-valor): p-valor del Test de Jaque-Bera.
- Skew: Coeficiente de Simetría de pearson de los residuos.
- Kurtosis: Coeficiente de apuntamiento de los residuos.

    - Analisis de los resultados:

        - Para que aceptemos la hipótesis de la normalidad nula (Chi-cuadrado y Jarque-Bera), nuestro p-valor debe ser mayor que alfa (0.05).
        - Para que aceptemos la hipótesis de la simetría de skew, nuestro valor tiene que ser próximo a 0.
        - Para que aceptemos la hipótesis de kurtosis, el valor debe ser próximo a 4.

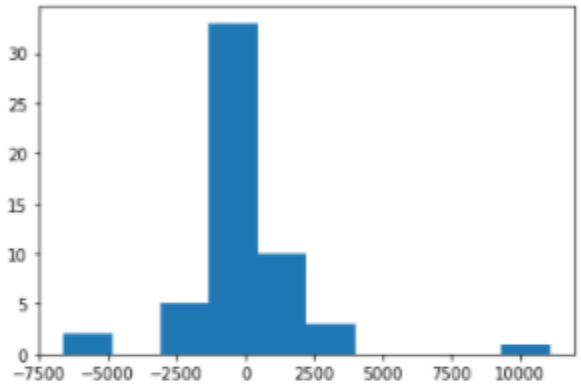
```
import statsmodels.stats.api as sms
name = ['Jarque-Bera', 'Chi^2 two-tail prob.', 'Skew', 'Kurtosis']
test = sms.jarque_bera(mco1.resid)
for i in range(4):
    print(name[i], test[i])
```

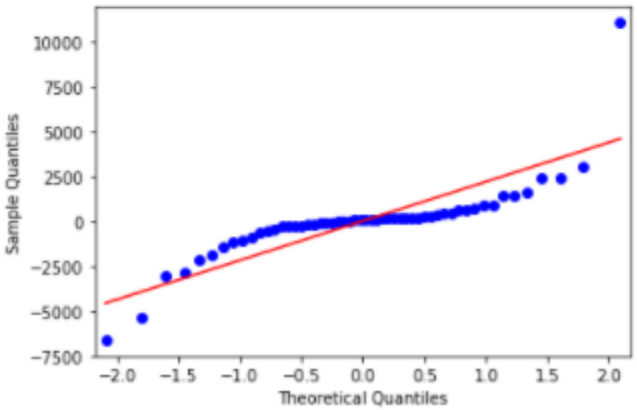
```
Jarque-Bera 323.6725265636269
Chi^2 two-tail prob. 5.192827366266701e-71
Skew 1.5756829717965635
Kurtosis 14.572525607665003
```

- Como nuestro Chi-cuadrado es: 5.192827366266701e-71 < 0.05, no podemos aceptar la hipótesis de la normalidad nula.
- El valor de nuestro Skew es: 1.5756829717965635, no podemos decir que es sigue un modelo simétrico ya que es no es próximo a 0.
- Nuestro valor de kurtosis es: 14.572525607665003, por lo que deberías rechazarlo ya que no es próximo a 4.

- Graficas:

```
from matplotlib import pyplot
from statsmodels.graphics.gofplots import qqplot
pyplot.hist(mco1.resid)
pyplot.show()
qqplot(mco1.resid, line='s')
pyplot.show()
```





Kolmogorov-Smirnov: Test de Hipótesis para contrastar si una muestra proviene de una distribución (en este caso normal):

```
: import statsmodels.stats.diagnostic as diag

  diag.kstest_normal(mco1.resid)

: (0.21227051059921132, 0.0009999999999998899)
```

Como nuestro p-valor: 0.0009999999999998899 < 0.05, rechazamos la hipótesis.

Tercer Modelo de nuestro proyecto (eliminando la variable Total Cases y Deaths 1mil/ poblation)

Primero podemos observar el summary:

OLS Regression Results

Dep. Variable:	y	R-squared:	0.968
Model:	OLS	Adj. R-squared:	0.964
Method:	Least Squares	F-statistic:	238.4
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	1.90e-33
Time:	17:01:51	Log-Likelihood:	-492.59
No. Observations:	54	AIC:	999.2
Df Residuals:	47	BIC:	1013.
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	142.7982	513.117	0.278	0.782	-889.461	1175.057
Total Recovered	0.0429	0.003	13.862	0.000	0.037	0.049
Total Tests	-0.0019	0.000	-4.079	0.000	-0.003	-0.001
Active Cases	0.0315	0.023	1.374	0.176	-0.015	0.078
Total Cases/1 mil population	-0.0074	0.012	-0.624	0.536	-0.031	0.016
Tests/1 mil population	-0.0010	0.003	-0.344	0.732	-0.007	0.005
Population	9.306e-06	1.08e-05	0.860	0.394	-1.25e-05	3.11e-05

Omnibus:	44.575	Durbin-Watson:	2.072
Prob(Omnibus):	0.000	Jarque-Bera (JB):	366.030
Skew:	1.793	Prob(JB):	3.29e-80
Kurtosis:	15.240	Cond. No.	7.09e+07

- En este modelo, tenemos un R-squared bastante aceptable con un valor: 0.968.

- También podemos ver que es un modelo significativo o fiable porque su Prob (F-statistic): 1.90e-33, esto quiere decir que cuanto más bajo sea el valor más fiable.

Test de linealidad:

- Test de Harvey-Collier:

- Estos test nos permite saber si un modelo sigue una disposición lineal, a la hora de diagnosticar los valores de los residuos.

```
from scipy import stats
print("Test de Harvey-Collier: ")
skip = len(mco1.params)  # bug in linear_harvey_collier
rr = sms.recursive_olsresiduals(mco1, skip=skip, alpha=0.95, order_by=None)
stats.ttest_1samp(rr[3][skip:], 0)
```

✓ 0.3s

```
Test de Harvey-Collier:

Ttest_1sampResult(statistic=0.5616104066420529, pvalue=0.5771076508615598)
```

Como el valor de p es: 0.5771076508615598 al ser mayor > 0.05, aceptamos la hipótesis.

- Test de RESET de Ramsey:

```
print("Test de RESET de Ramsey: ")
rr=oi.reset_ramsey(mco1, degree=2)
rr
```

✓ 0.4s

```
Test de RESET de Ramsey:

<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=array([[6.18544145]]), p=0.016569696623412785, df_denom=46, df_num=1>
```

Como el valor de p es extremadamente pequeño no podemos aceptar la hipótesis ya que esta debería ser mayor que 0.05

Test de Autocorrelación:

- Test de Durbin Watson:

1.- Si el valor es 0, existe una correlación bastante fuerte (positivamente).
2.- Si el valor es 4, existe una correlación bastante fuerte (negativamente).
3.- Si está cerca de 2, quiere decir que no existe correlación.

4.- Miramos en la tabla de DW para comprobar su valor.

```
from statsmodels.stats.stattools import durbin_watson
dw=durbin_watson(mco1.resid)

print("Durbin-Watson:", dw)

rho= 1 - dw/2 # ro estimado inicial
print ("Rho inicial: ", rho) # rho inicial

mco_autocorr=sm.GLSAR(Y, sm.add_constant(X), rho=rho)
res = mco_autocorr.iterative_fit(maxiter=1000, rtol = 10**(-10)) # iteraciones y

print ('Iterations used = %d Converged %s' % (res.iter, res.converged) )
print ('Rho = ', mco_autocorr.rho)

res.summary()
```

✓ 0.1s

```
Durbin-Watson: 2.0722241441761406
Rho inicial:  -0.036112072088070324
Iterations used = 17 Converged True
Rho =  [-0.08050849]
```

GLSAR Regression Results

Dep. Variable:	y	R-squared:	0.969
Model:	GLSAR	Adj. R-squared:	0.965
Method:	Least Squares	F-statistic:	239.8
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	5.57e-33
Time:	17:01:52	Log-Likelihood:	-483.17
No. Observations:	53	AIC:	980.3
Df Residuals:	46	BIC:	994.1
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	148.6552	495.493	0.300	0.766	-848.720	1146.030
Total Recovered	0.0443	0.003	13.744	0.000	0.038	0.051
Total Tests	-0.0021	0.000	-4.334	0.000	-0.003	-0.001
Active Cases	0.0460	0.027	1.708	0.094	-0.008	0.100
Total Cases/1 mil population	-0.0078	0.012	-0.656	0.515	-0.032	0.016
Tests/1 mil population	-0.0012	0.003	-0.424	0.674	-0.007	0.005
Population	1.256e-05	1.08e-05	1.159	0.252	-9.25e-06	3.44e-05

Omnibus:	38.380	Durbin-Watson:	1.977
Prob(Omnibus):	0.000	Jarque-Bera (JB):	284.493
Skew:	1.492	Prob(JB):	1.67e-62
Kurtosis:	13.951	Cond. No.	7.00e+07

Al ser un valor próximo al 2, podemos aceptar que este modelo. Residuos incorrelados:

DU < DW < 4 - DU

Podemos comentar que para aceptar este modelo la aproximación que se realiza entre el Rho inicial y el Rho final debe ser pequeña.

- Test de Ljung-Box:

1.- Con este test podemos probar si existe relación en mi perturbación aleatoria con la de x años antes.
2.- Los lags son los retardos, pudiendo poner cualquier número de años.
3.- Explicación: da el p valor, que es la siguiente lista, rechazo las hipótesis si los p-valores son extremadamente chicos, es decir existe relación con los otros años.

```
from statsmodels.stats.diagnostic import acorr_ljungbox # analizamos con Ljung-Box s
mco2=sm.OLS(Y, sm.add_constant(X)).fit()
acorr_ljungbox(mco2.resid, lags=3) # lags son los retardos, comparo hasta con 3 años

#
```

✓ 0.3s

```
D:\Anaconda\lib\site-packages\statsmodels\stats\diagnostic.py:559: FutureWarning: The va
return a DataFrame now.  Set return_df to False to silence this warning.
  warnings.warn(msg, FutureWarning)
(array([0.14616019, 0.16140398, 0.74233923]),
 array([0.70223216, 0.92246855, 0.86320299]))
```

Como los valores del primer array con respectos a los del segundo array, en nuestro caso, como son valores no extremadamente chicos podemos aceptar la hipótesis.

Test de Heterocedasticidad:

En un modelo hay heterocedasticidad cuando la varianza de los errores no es igual en todas las observaciones realizadas.

Por lo que buscamos comprobar así, si se cumple uno de los requisitos básicos de las hipótesis de los modelos lineales.

- Test de goldfelquandt (test para muestra pequeñas):

- Split es el número de elementos que pone en cada trozo, si no pones nada divide en 2 por defecto.
- Para dividir entre 3 las variables, para comparar los maximos y minimos.

```
import statsmodels.stats.api as sms

GQ = sms.het_goldfeldquandt(mco1.resid, mco1.model.exog)

#Hipotesis nula homocedasticidad, no tengo un problema de heterocedasticidad
#Todo numero mayor que 0.05 rechazo entonces tengo heterocedasticidad

print (GQ)
```

✓ 0.3s

```
(1.110125134092535, 0.40879115823431933, 'increasing')
```

Este test nos sirve para modelos con pocos datos, como es nuestro caso, podemos observar que nuestro p-valor es aceptable ya que supera el valor de 0.05, por lo que aceptamos la hipótesis nula de homocedasticidad.

p-valor: 0.40879115823431933 > 0.05

- Test de Breush-Pagan (test para muestras grandes):

- Los dos últimos valores me interesan , el estadístico experimental y P valor:
  - (como este es mayor que 0.05 no tengo problema de heterocedasticidad) tengo homocedasticidad porque nuestro valor es más pequeño 0,05.
  - (alpha nivel de significación a partir del cual empiezo a rechazar el modelo).

```
#BREUSH-PAGAN
BP = sms.het_breuschpagan(mco1.resid, mco1.model.exog)


print (BP)
```

✓ 0.4s

```
(16.606630219096804, 0.010843026401722437, 3.478832499231621, 0.006296061709458417)
```

Estos test son para modelos con un número significativo de muestras, aun así nuestro p-valor sigue siendo muy bajo por lo que tenemos problemas de heterocedasticidad, al rechazar la hipótesis nula.

p-valor: 0.010843026401722437 < 0.05

- Test de White:

- El test de white es muy parecido, pero tiene en cuenta el producto de las variables.
- Si me da significación global no tengo problema de heterocedasticidad.

```
#WHITE
W=sms.het_white(mco1.resid, mco1.model.exog)
print (W)
```

✓ 0.4s

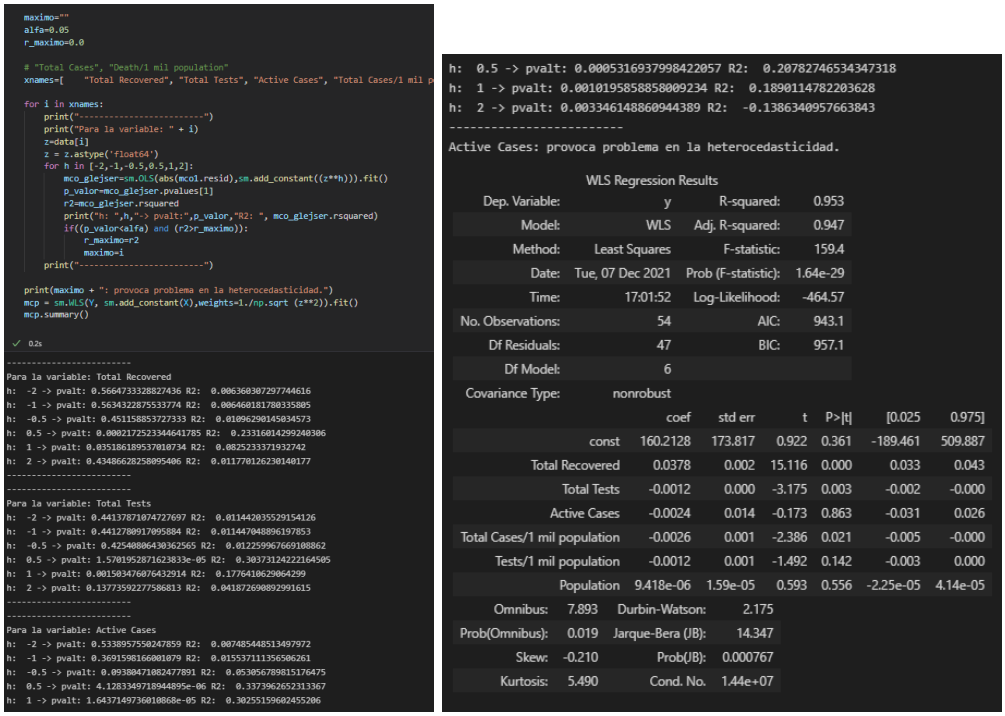
```
(49.8381138913824, 0.0022312488984882976, 13.411872910882915, 7.308741791044887e-10)
```

En este caso, pasa lo mismo, como es para muestras grandes no nos sirve y no da un valor no aceptable:

p-valor: 7.308741791044887e-10 < 0.05.

- Test de Glejser:

- Vamos a comprobar todas estas variables y nos vamos a quedar con la que tenga el R2 más grande, si este valor es mayor que 0.05.
- Ese será nuestro Z^2, el cual se puede corregir con mínimos cuadrados ponderados, multiplicando mis datos por 1/sqrt(Z^2).

- 

```
maximo=""
alfa=0.05
r_maximo=0.0

# "Total Cases", "Death/1 mil population"
xnames=[    "Total Recovered", "Total Tests", "Active Cases", "Total Cases/1 mil p

for i in xnames:
    print("-------------------------")
    print("Para la variable: " + i)
    z=data[i]
    z = z.astype('float64')
    for h in [-2,-1,-0.5,0.5,1,2]:
        mco_glejser=sm.OLS(abs(mco1.resid),sm.add_constant((z**h))).fit()
        p_valor=mco_glejser.pvalues[1]
        r2=mco_glejser.rsquared
        print("h: ",h,"-> pvalt:",p_valor,"R2: ", mco_glejser.rsquared)
        if((p_valor<alfa) and (r2>r_maximo)):
            r_maximo=r2
            maximo=i
    print("-------------------------")

print(maximo + ": provoca problema en la heterocedasticidad.")
mcp = sm.WLS(Y, sm.add_constant(X),weights=1./np.sqrt (z**2)).fit()
mcp.summary()
```

✓ 0.2s

```
-------------------------
Para la variable: Total Recovered
h:  -2 -> pvalt: 0.5664733328827436 R2:  0.006360307297744616
h:  -1 -> pvalt: 0.5634322875533774 R2:  0.006460181780335805
h:  -0.5 -> pvalt: 0.451158853727333 R2:  0.01096290145034573
h:  0.5 -> pvalt: 0.0002172523344641785 R2:  0.2331601429924030 6
h:  1 -> pvalt: 0.035186189537010734 R2:  0.0825233371932742
h:  2 -> pvalt: 0.43486628258095406 R2:  0.01177012623014017 7
-------------------------
-------------------------
Para la variable: Total Tests
h:  -2 -> pvalt: 0.44137871074727697 R2:  0.011442035529154126
h:  -1 -> pvalt: 0.4412780917095884 R2:  0.011447048961978 53
h:  -0.5 -> pvalt: 0.42540806430362565 R2:  0.012259967669108862
h:  0.5 -> pvalt: 1.5701952871623833e-05 R2:  0.3037312422216 5405
h:  1 -> pvalt: 0.0015034760764329 14 R2:  0.17764106290642 99
h:  2 -> pvalt: 0.13773592277586813 R2:  0.041872690892991615
-------------------------
-------------------------
Para la variable: Active Cases
h:  -2 -> pvalt: 0.5338957550247859 R2:  0.007485448513497972
h:  -1 -> pvalt: 0.3691598166001079 R2:  0.015537111356506261
h:  -0.5 -> pvalt: 0.09380471082477891 R2:  0.05305678981517 6475
h:  0.5 -> pvalt: 4.1283349718944895e-06 R2:  0.337396265231 3367
h:  1 -> pvalt: 1.6437149736010868e-05 R2:  0.30255159602455206
```

```
h:  0.5 -> pvalt: 0.0005316937998422057 R2:  0.20782746534347318
h:  1 -> pvalt: 0.0010195858858009234 R2:  0.1890114782203628
h:  2 -> pvalt: 0.003346148860944389 R2:  -0.1386340957663843
-------------------------
Active Cases: provoca problema en la heterocedasticidad.
```

WLS Regression Results

Dep. Variable:	y	R-squared:	0.953
Model:	WLS	Adj. R-squared:	0.947
Method:	Least Squares	F-statistic:	159.4
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	1.64e-29
Time:	17:01:52	Log-Likelihood:	-464.57
No. Observations:	54	AIC:	943.1
Df Residuals:	47	BIC:	957.1
Df Model:	6		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	160.2128	173.817	0.922	0.361	-189.461	509.887
Total Recovered	0.0378	0.002	15.116	0.000	0.033	0.043
Total Tests	-0.0012	0.000	-3.175	0.003	-0.002	-0.000
Active Cases	-0.0024	0.014	-0.173	0.863	-0.031	0.026
Total Cases/1 mil population	-0.0026	0.001	-2.386	0.021	-0.005	-0.000
Tests/1 mil population	-0.0012	0.001	-1.492	0.142	-0.003	0.000
Population	9.418e-06	1.59e-05	0.593	0.556	-2.25e-05	4.14e-05

Omnibus:	7.893	Durbin-Watson:	2.175
Prob(Omnibus):	0.019	Jarque-Bera (JB):	14.347
Skew:	-0.210	Prob(JB):	0.000767
Kurtosis:	5.490	Cond. No.	1.44e+07

Test de Multicolinealidad:

- Condition number:

```
print("Condition number: ", mco1.condition_number) #Número de Condición

#Tendriamos problemas de multicolinealidad si fuese mayor que 900, ya que esta al cuadrado
# Tenemos un problema gordo porque nuestro valor > 900.
```

✓ 0.2s

```
Condition number:  70852017.19199444
```

Nuestro condition number es altísimo, lo que quiere decir que nuestro modelo presenta variables que dependen unas de otras.

Condition number: 70852017.19199444 s> 900

Como nuestro condition number es mayor que 900 tenemos una dependencia entre variables y un problema grave.

- Test de Vifs:

- Comprobamos el factor de inflación de la varianza en todas las variables.
    - Posibles resultados:
        - Si sale mayor que 10: tenemos problema de multicolinealidad.
        - Si sale mayor que 20: problema gordo de multicolinealidad.

```
import statsmodels.stats.outliers_influence as oi

vifs=[oi.variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

vifs
```

✓ 0.4s

```
[16.313671798394967,
 19.945690886989045,
 1.4226112050396271,
 1.7864186151145003,
 2.2063233044080075,
 1.721470504232903]
```

En este test, por lo tanto, se ve que hemos conseguido disminuir la dependencia entre las variables, pues ahora, solo hay problemas de correlación con 2 variables.

- Matriz de Correlaciones:

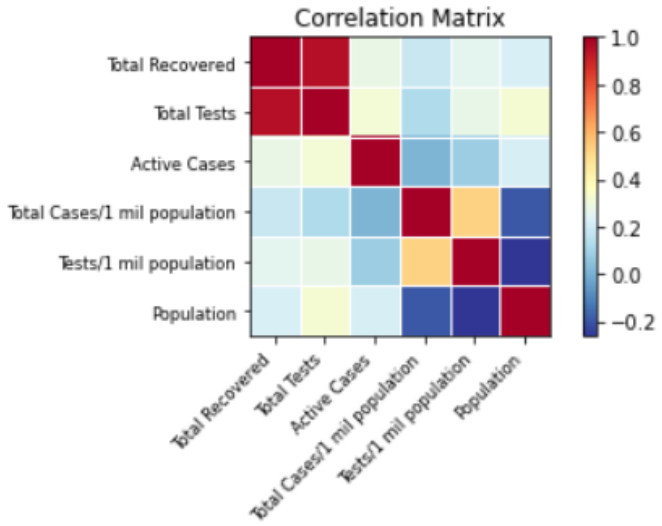
- Se busca que la matriz se parezca a la matriz identidad para sacar como factor común la varianza como la multiplicación de esta por la identidad.

```
#Matriz de correlacciones
corr_matrix=np.corrcoef(X.T) #X.T = matriz X traspuesta
print(corr_matrix)
```

✓ 0.3s

```
[[ 1.          0.95771343  0.28327872  0.18781035  0.26473752  0.23077628]
 [ 0.95771343  1.          0.32002308  0.1293295   0.27776218  0.32629645]
 [ 0.28327872  0.32002308  1.          0.01973277  0.08419675  0.22591983]
 [ 0.18781035  0.1293295   0.01973277  1.          0.52916249 -0.19110093]
 [ 0.26473752  0.27776218  0.08419675  0.52916249  1.         -0.26075754]
 [ 0.23077628  0.32629645  0.22591983 -0.19110093 -0.26075754  1.        ]]
```

Nuestra matriz se asemeja a la matriz de identidad, en cierto sentido, por lo que podemos decir que si se cumple lo que sea que se tiene que cumplir.



Se ve reflejado en el gráfico la mejora de la que hablábamos antes, pues ahora las variables no dependen tanto entre sí.

- Normalidad de los Residuos:

- Jarque-Bera: Test de hipótesis que contrasta si los datos de la muestra tienen el coeficiente de simetría y la curtósis de una distribución normal.
- chi^2 (p-valor): p-valor del Test de Jaque-Bera.
- Skew: Coeficiente de Simetría de pearson de los residuos.
- Kurtosis: Coeficiente de apuntamiento de los residuos.

- Análisis de los resultados:
    - Para que aceptemos la hipótesis de la normalidad nula (Chi-cuadrado y Jarque-Bera), nuestro p-valor debe ser mayor que alfa (0.05).
    - Para que aceptemos la hipótesis de la simetría de skew, nuestro valor tiene que ser próximo a 0.
    - Para que aceptemos la hipótesis de kurtosis, el valor debe ser próximo a 4.

```
mco_res = sm.OLS(Y, sm.add_constant(X)).fit()
name = ['Jarque-Bera', 'Chi^2 two-tail prob.', 'Skew', 'Kurtosis']
test = sms.jarque_bera(mco_res.resid)
for i in range(4):
    print(name[i], test[i])
```

✓ 0.1s

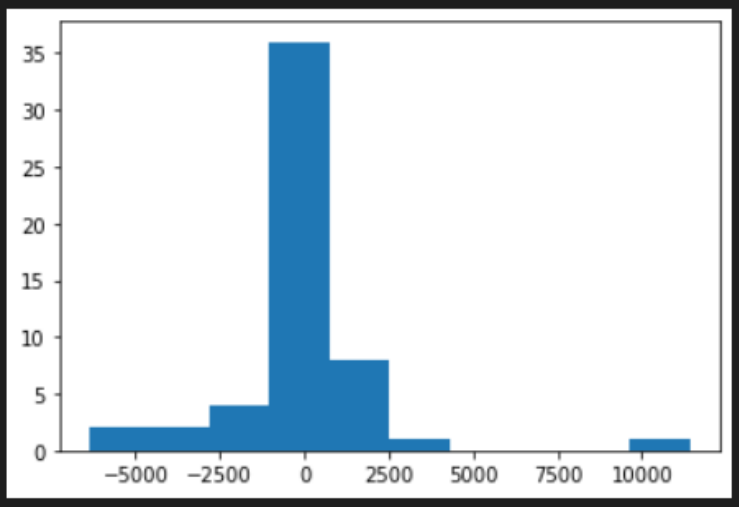
```
Jarque-Bera 366.03037511520256
Chi^2 two-tail prob. 3.292410205817151e-80
Skew 1.7933206109034512
Kurtosis 15.239941644628184
```

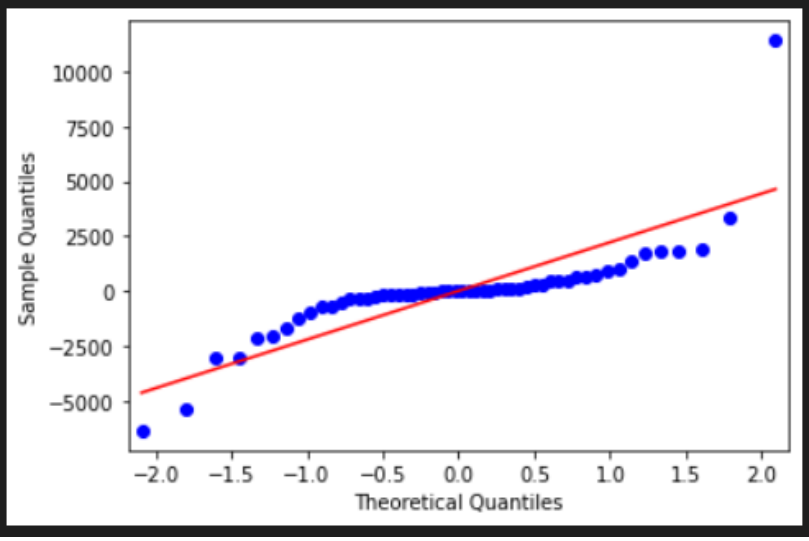
- Como nuestro Chi-cuadrado es: 3.292410205817151e-80 < 0.05, no podemos aceptar la hipótesis de la normalidad nula.
- El valor de nuestro Skew es: 1.7933206109034512, no podemos decir que es sigue un modelo simétrico ya que es no es próximo a 0.
- Nuestro valor de kurtosis es: 15.239941644628184, por lo que deberías rechazarlo ya que no es próximo a 4.

- Gráficas:

```
from matplotlib import pyplot
from statsmodels.graphics.gofplots import qqplot
pyplot.hist(mco1.resid)
pyplot.show()
qqplot(mco1.resid, line='s')
pyplot.show()
```

✓ 0.5s





Kolmogorov-Smirnov: Test de Hipótesis para contrastar si una muestra proviene de una distribución (en este caso normal):

```
import statsmodels.stats.diagnostic as diag

diag.kstest_normal(mco1.resid)
```

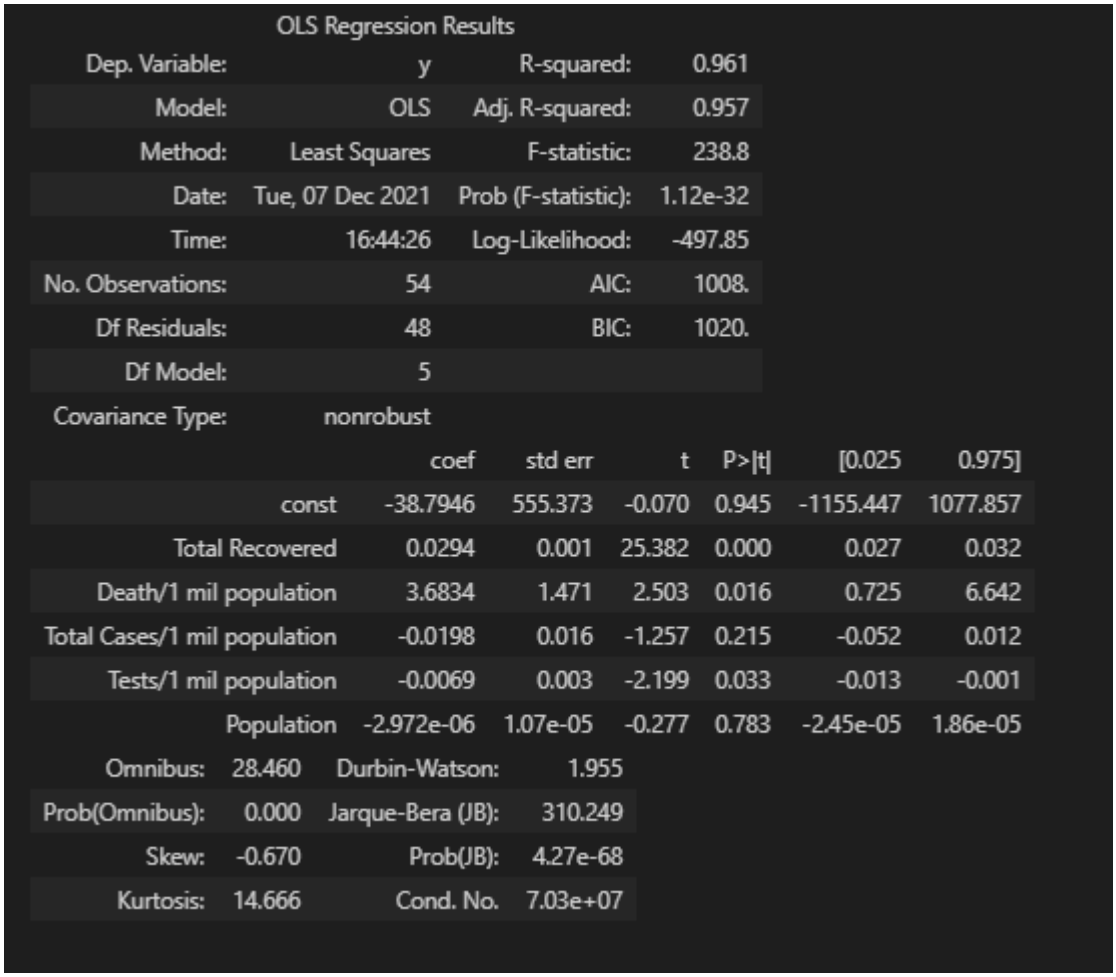
✓ 0.2s

```
(0.2110380553651592, 0.0009999999999998899)
```

Como nuestro p-valor: 0.0009999999999998899 < 0.05, rechazamos la hipótesis.

Modelo final del proyecto:

(Eliminamos las variables "Total Cases", "Total Tests", "Active Cases")
Una vez cargado el modelo, vamos a ver los resultados de regresión:



OLS Regression Results

Dep. Variable:	y	R-squared:	0.961
Model:	OLS	Adj. R-squared:	0.957
Method:	Least Squares	F-statistic:	238.8
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	1.12e-32
Time:	16:44:26	Log-Likelihood:	-497.85
No. Observations:	54	AIC:	1008.
Df Residuals:	48	BIC:	1020.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-38.7946	555.373	-0.070	0.945	-1155.447	1077.857
Total Recovered	0.0294	0.001	25.382	0.000	0.027	0.032
Death/1 mil population	3.6834	1.471	2.503	0.016	0.725	6.642
Total Cases/1 mil population	-0.0198	0.016	-1.257	0.215	-0.052	0.012
Tests/1 mil population	-0.0069	0.003	-2.199	0.033	-0.013	-0.001
Population	-2.972e-06	1.07e-05	-0.277	0.783	-2.45e-05	1.86e-05

Omnibus:	28.460	Durbin-Watson:	1.955
Prob(Omnibus):	0.000	Jarque-Bera (JB):	310.249
Skew:	-0.670	Prob(JB):	4.27e-68
Kurtosis:	14.666	Cond. No.	7.03e+07

- En este modelo, tenemos un R-squared bastante aceptable con un valor: 0.961.

- También podemos ver que es un modelo significativo o fiable porque su Prob (F-statistic): 1.12e-32, esto quiere decir que cuanto más bajo sea el valor más fiable.

Test de linealidad:

- Test de Harvey-Collier:

- Estos test nos permite saber si un modelo sigue una disposición lineal, a la hora de diagnosticar los valores de los residuos.

```
from scipy import stats
skip = len(mco1.params)  # bug in linear_harvey_collier
rr = sms.recursive_olsresiduals(mco1, skip=skip, alpha=0.95, order_by=None)
stats.ttest_1samp(rr[3][skip:], 0)
```

✓ 0.9s

```
Ttest_1sampResult(statistic=0.06295879195233976, pvalue=0.950066315053199)
```

Como el valor de p es: 0.950066315053199 al ser mayor > 0.05, aceptamos la hipótesis.

- Test de RESET de Ramsey:

```
print("Test de RESET de Ramsey: ")
rr=oi.reset_ramsey(mco1, degree=2)
rr
```

✓ 0.3s

```
Test de RESET de Ramsey:
<class 'statsmodels.stats.contrast.ContrastResults'>
<F test: F=array([[27.93605752]]), p=3.184640934567803e-06, df_denom=47, df_num=1>
```

Como podemos ver que el p-valor en nuestro caso es de p=3.184640934567803e-06, podemos decir que no aceptamos la hipótesis porque su valor es menor que 0.05.

Test de Autocorrelación:

- Test de Durbin Watson:

1.- Si el valor es 0, existe una correlación bastante fuerte (positivamente).
2.- Si el valor es 4, existe una correlación bastante fuerte (negativamente).
3.- Si está cerca de 2, quiere decir que no existe correlación.
4.- Miramos en la tabla de DW para comprobar su valor.

```
from statsmodels.stats.stattools import durbin_watson
dw=durbin_watson(mco1.resid)

print("Durbin-Watson:", dw)

rho= 1 - dw/2 # ro estimado inicial
print ("Rho inicial: ", rho) # rho inicial

mco_autocorr=sm.GLSAR(Y, sm.add_constant(X), rho=rho)
res = mco_autocorr.iterative_fit(maxiter=1000, rtol = 10**(-10)) # iteraciones y tolerancia, 1

print ('Iterations used = %d Converged %s' % (res.iter, res.converged) )
print ('Rho = ', mco_autocorr.rho)

res.summary()
```

✓ 0.1s

```
Durbin-Watson: 1.9549380214767726
Rho inicial:  0.022530989261613676
Iterations used = 12 Converged True
Rho =  [0.02530359]
```

GLSAR Regression Results

Dep. Variable:	y	R-squared:	0.962
Model:	GLSAR	Adj. R-squared:	0.958
Method:	Least Squares	F-statistic:	236.6
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	4.22e-32
Time:	16:44:27	Log-Likelihood:	-488.90
No. Observations:	53	AIC:	989.8
Df Residuals:	47	BIC:	1002.
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-72.2297	567.355	-0.127	0.899	-1213.602	1069.142
Total Recovered	0.0294	0.001	25.226	0.000	0.027	0.032
Death/1 mil population	3.6520	1.479	2.468	0.017	0.676	6.628
Total Cases/1 mil population	-0.0199	0.016	-1.257	0.215	-0.052	0.012
Tests/1 mil population	-0.0067	0.003	-2.090	0.042	-0.013	-0.000
Population	-3.661e-06	1.08e-05	-0.338	0.736	-2.54e-05	1.81e-05

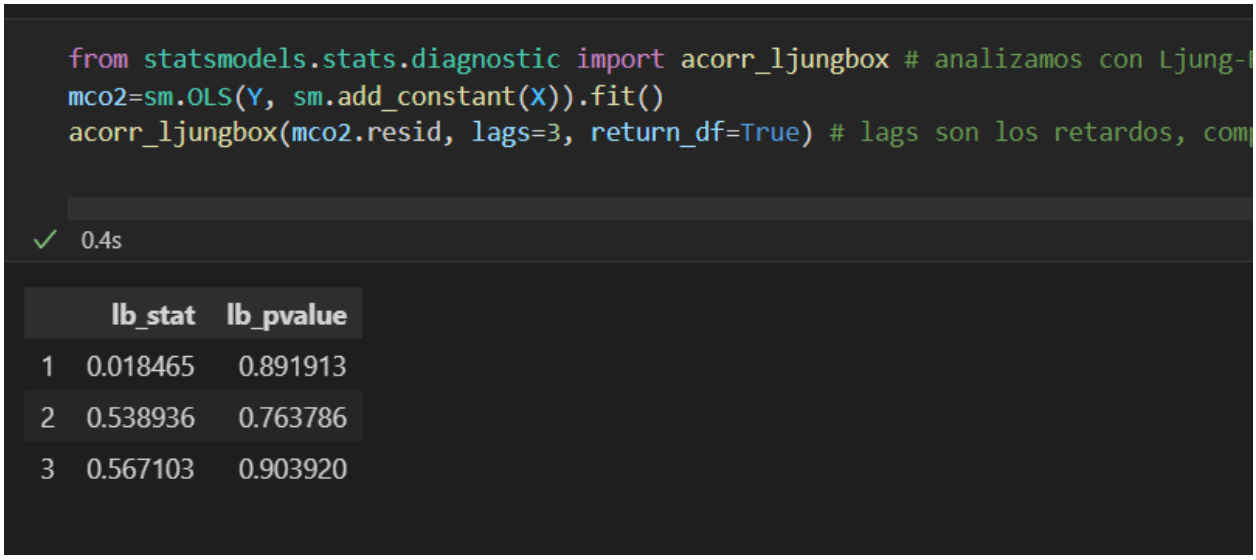
Omnibus:	28.324	Durbin-Watson:	2.007
Prob(Omnibus):	0.000	Jarque-Bera (JB):	320.830
Skew:	-0.648	Prob(JB):	2.15e-70
Kurtosis:	14.983	Cond. No.	7.01e+07

Al ser un valor próximo al 2, podemos aceptar este modelo. Los residuos siguen siendo incorrelados, ya que: DU < DW < 4 - DU.

Podemos comentar que para aceptar este modelo la aproximación que se realiza entre el Rho inicial y el Rho final debe ser pequeña.

- Test de Ljung-Box:

1.- Con este test podemos probar si existe relación en mi perturbación aleatoria con la de x años antes.
2.- Los lags son los retardos, pudiendo poner cualquier número de años.
3.- Explicación: da el p valor, que es la siguiente lista, acepto las 3 hipótesis ya que todos los p valores son más grande que la alfa (0.05).
4.- No deberíamos tener problema de autocorrelación.



```
from statsmodels.stats.diagnostic import acorr_ljungbox # analizamos con Ljung-
mco2=sm.OLS(Y, sm.add_constant(X)).fit()
acorr_ljungbox(mco2.resid, lags=3, return_df=True) # lags son los retardos, com
```

✓ 0.4s

	lb_stat	lb_pvalue
1	0.018465	0.891913
2	0.538936	0.763786
3	0.567103	0.903920

Como los valores del primer array con respectos a los del segundo array, en nuestro caso, como son valores no extremadamente chicos podemos aceptar la hipótesis.

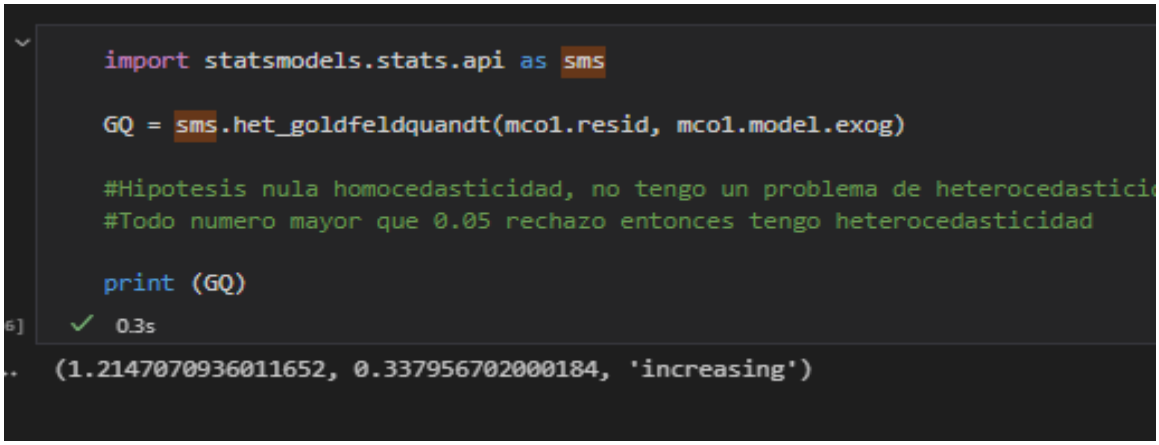
Test de Heterocedasticidad:

En un modelo hay heterocedasticidad cuando la varianza de los errores no es igual en todas las observaciones realizadas.

Por lo que buscamos comprobar así, si se cumple uno de los requisitos básicos de las hipótesis de los modelos lineales.

- Test de goldfelquandt (test para muestra pequeñas):

- Split es el número de elementos que pone en cada trozo, si no pones nada divide en 2 por defecto.
- Para dividir entre 3 las variables, para comparar los maximos y minimos.

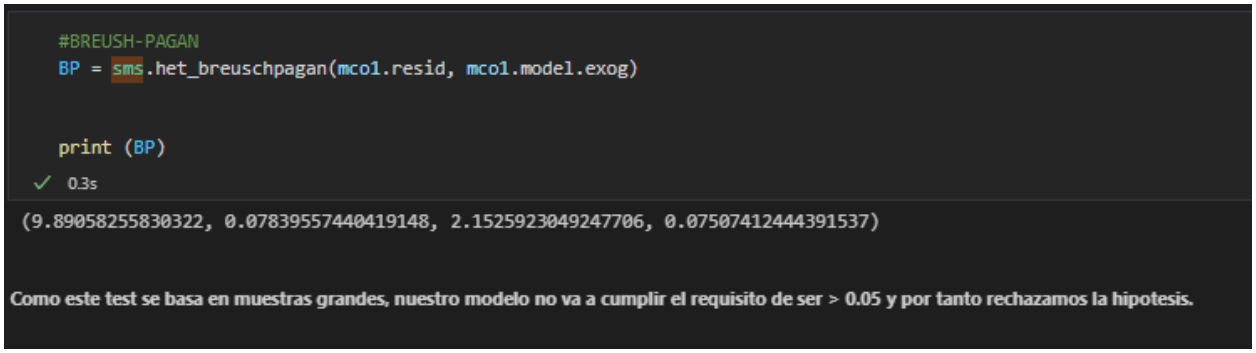


Este test nos sirve para modelos con pocos datos, como es nuestro caso, podemos observar que nuestro p-valor es aceptable ya que supera el valor de 0.05, por lo que aceptamos la hipótesis nula de homocedasticidad.

p-valor: 0.337956702000184 > 0.05.

- Test de Breush-Pagan (test para muestras grandes):

- Los dos últimos valores me interesan , el estadístico experimental y P valor:
  - (como este es mayor que 0.05 no tengo problema de heterocedasticidad) tengo homocedasticidad porque nuestro valor es más pequeño 0,05.
  - (alpha nivel de significación a partir del cual empiezo a rechazar el modelo).

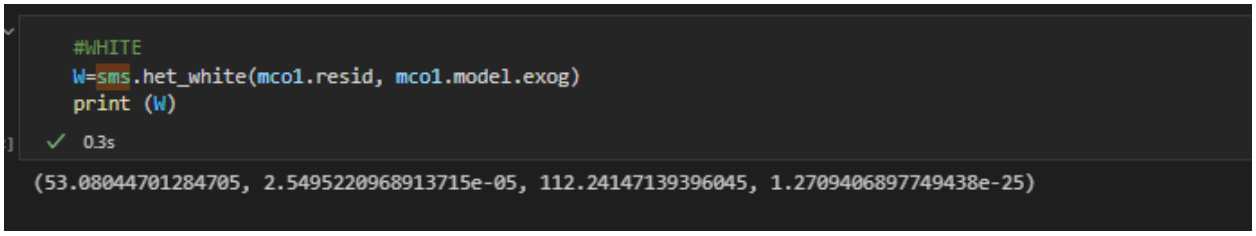


Como este nuestro modelo sale con un valor de: 0.07507412444391537 > 0.05, por tanto aceptamos la hipótesis y decimos que no tenemos problemas de heterocedasticidad.

P-valor: 0.07507412444391537 < 0.05

- Test de White:

- El test de white es muy parecido, pero tiene en cuenta el producto de las variables.
- Si me da significación global no tengo problema de heterocedasticidad.



En este caso, como es para muestras grandes no nos sirve y no da un valor no aceptable:

p-valor: 3.642084309485573e-30 < 0.05.

Por tanto rechazamos la hipotesis.

- Test de Glejser:

- Vamos a comprobar todas estas variables y nos vamos a quedar con la que tenga el R2 más grande, si este valor es mayor que 0.05.
- Ese será nuestro Z^2, el cual se puede corregir con mínimos cuadrados ponderados, multiplicando mis datos por 1/sqrt(Z^2).

```
maximo=""
alfa=0.05
r_maximo=0.0

# "Total Cases", "Total Tests", "Active Cases",
xnames=[    "Total Recovered", "Death/1 mil population", "Total Cases/1 mil population", "Test

for i in xnames:
    print("------------------------")
    print("Para la variable: " + i)
    z=data[i]
    z = z.astype('float64')
    for h in [-2,-1,-0.5,0.5,1,2]:
        mco_glejser=sm.OLS(abs(mco1.resid),sm.add_constant((z**h))).fit()
        p_valor=mco_glejser.pvalues[1]
        r2=mco_glejser.rsquared
        print("h: ",h,"-> pvalt:",p_valor,"R2: ", mco_glejser.rsquared)
        if((p_valor<alfa) and (r2>r_maximo)):
            r_maximo=r2
            maximo=i
    print("------------------------")

print(maximo + ": provoca problema en la heterocedasticidad.")
mcp = sm.WLS(Y, sm.add_constant(X),weights=1./np.sqrt (z**2)).fit()
mcp.summary()
```

✓ 0.2s

```
------------------------
Para la variable: Total Recovered
h:  -2 -> pvalt: 0.876913072681035 R2:  0.00046567395254559774
h:  -1 -> pvalt: 0.8742276214470905 R2:  0.00048638267421252657
h:  -0.5 -> pvalt: 0.7572246925240808 R2:  0.0018537174510683263
h:  0.5 -> pvalt: 6.018350817537882e-06 R2:  0.32804860328781416
h:  1 -> pvalt: 0.0010046247808382911 R2:  0.18944214657600844
h:  2 -> pvalt: 0.0631749677198063 R2:  0.06482583513830187
------------------------
------------------------
Para la variable: Death/1 mil population
h:  -2 -> pvalt: 0.4274594312633496 R2:  0.012152358419017961
h:  -1 -> pvalt: 0.11498308049665701 R2:  0.04709098651233212
h:  -0.5 -> pvalt: 0.03708483120684973 R2:  0.08092591862347498
h:  0.5 -> pvalt: 0.11728095398125604 R2:  0.0465154583583931
h:  1 -> pvalt: 0.3396570490095645 R2:  0.017546551541621103
h:  2 -> pvalt: 0.7497598533679334 R2:  0.001973351251093658
------------------------
------------------------
Para la variable: Total Cases/1 mil population
h:  -2 -> pvalt: 0.3243927733717046 R2:  0.018679499841142477
h:  -1 -> pvalt: 0.14217808486819083 R2:  0.04096359446196163
h:  -0.5 -> pvalt: 0.06871621766396395 R2:  0.06230630356933164 5
h:  0.5 -> pvalt: 0.21616208144056578 R2:  0.02926348695392722
h:  1 -> pvalt: 0.5923351743488876 R2:  0.0055514731106768 3
```

```
h:  0.5 -> pvalt: 0.0039027467123639572 R2:  0.1492951623403913
h:  1 -> pvalt: 0.005676316059147533 R2:  0.13802111614657986
h:  2 -> pvalt: 0.009355853945953976 R2:  -0.18507264437581172
------------------------
Total Recovered: provoca problema en la heterocedasticidad.
```

WLS Regression Results

Dep. Variable:	y	R-squared:	0.946
Model:	WLS	Adj. R-squared:	0.940
Method:	Least Squares	F-statistic:	167.4
Date:	Tue, 07 Dec 2021	Prob (F-statistic):	3.72e-29
Time:	16:44:27	Log-Likelihood:	-468.53
No. Observations:	54	AIC:	949.1
Df Residuals:	48	BIC:	961.0
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	94.2346	194.957	0.483	0.631	-297.752	486.222
Total Recovered	0.0295	0.001	22.708	0.000	0.027	0.032
Death/1 mil population	0.9318	0.536	1.737	0.089	-0.147	2.010
Total Cases/1 mil population	-0.0049	0.003	-1.941	0.058	-0.010	0.000
Tests/1 mil population	-0.0030	0.001	-3.618	0.001	-0.005	-0.001
Population	-7.075e-06	1.56e-05	-0.452	0.653	-3.85e-05	2.44e-05

Omnibus:	25.144	Durbin-Watson:	2.092
Prob(Omnibus):	0.000	Jarque-Bera (JB):	89.043
Skew:	-1.075	Prob(JB):	4.62e-20
Kurtosis:	8.912	Cond. No.	1.52e+07

Test de Multicolinealidad:

- Condition number:

```
print("Condition number: ", mco1.condition_number) #Número de Condición

#Tendriamos problemas de multicolinealidad si fuese mayor que 900, ya que esta al cuadrado
# Tenemos un problema gordo porque nuestro valor > 900.
```

✓ 0.4s

```
Condition number:  70268054.63283545
```

Nuestro condition number es altísimo, lo que quiere decir que nuestro modelo presenta variables que dependen unas de otras.

Condition number: 70268054.63283534 > 900

Como nuestro condition number es mayor que 900 tenemos una dependencia entre variables y un problema grave.

- Test de Vifs:

- Comprobamos el factor de inflación de la varianza en todas las variables.
    - Posibles resultados:
        - Si sale mayor que 10: tenemos problema de multicolinealidad.
        - Si sale mayor que 20: problema gordo de multicolinealidad.

```
import statsmodels.stats.outliers_influence as oi

vifs=[oi.variance_inflation_factor(X.values, i) for i in range(X.shape[1])]

vifs
```

✓ 0.7s

```
[2.681064354031329,
 2.127745109432254,
 4.23615399265927,
 1.892149893932364,
 1.171386713320542]
```

En este caso, como vemos, el VIF de todas las variables no supera en ningún caso los 20 puntos, por lo que ya no tenemos problemas grandes de multicolinealidad.

- Matriz de Correlaciones:

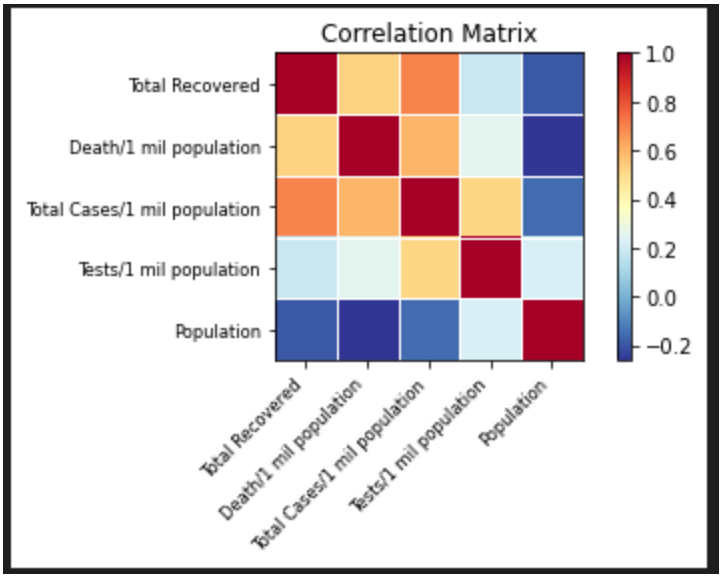
  - Se busca que la matriz se parezca a la matriz identidad para sacar como factor común la varianza como la multiplicación de esta por la identidad.

```
#Matriz de correlacciones
corr_matrix=np.corrcoef(X.T) #X.T = matriz X traspuesta
print(corr_matrix)
```

✓ 0.1s

```
[[ 1.          0.52916249  0.70020222  0.18781035 -0.19110093]
 [ 0.52916249  1.          0.59659099  0.26473752 -0.26075754]
 [ 0.70020222  0.59659099  1.          0.52177148 -0.15004533]
 [ 0.18781035  0.26473752  0.52177148  1.          0.23077628]
 [-0.19110093 -0.26075754 -0.15004533  0.23077628  1.        ]]
```

Nuestra matriz se asemeja a la matriz de identidad, en cierto sentido, por lo que podemos decir que si se cumple lo que sea que se tiene que cumplir.



Con este gráfico podemos ver las variables que muestran los mismos datos o que no son tan necesarios ya que la información que aportan es similar a la que muestra su relación.

- Normalidad de los Residuos:

- Jarque-Bera: Test de hipótesis que contrasta si los datos de la muestra tienen el coeficiente de simetría y la curtósis de una distribución normal.
- chi^2 (p-valor): p-valor del Test de Jaque-Bera.
- Skew: Coeficiente de Simetría de pearson de los residuos.
- Kurtosis: Coeficiente de apuntamiento de los residuos.

    - Analisis de los resultados:

        - Para que aceptemos la hipótesis de la normalidad nula (Chi-cuadrado y Jarque-Bera), nuestro p-valor debe ser mayor que alfa (0.05).
        - Para que aceptemos la hipótesis de la simetría de skew, nuestro valor tiene que ser próximo a 0.
        - Para que aceptemos la hipótesis de kurtosis, el valor debe ser próximo a 4.

```
name = ['Jarque-Bera', 'Chi^2 two-tail prob.', 'Skew', 'Kurtosis']
test = sms.jarque_bera(mco1.resid)
for i in range(4):
    print(name[i], test[i])
```

✓ 0.5s

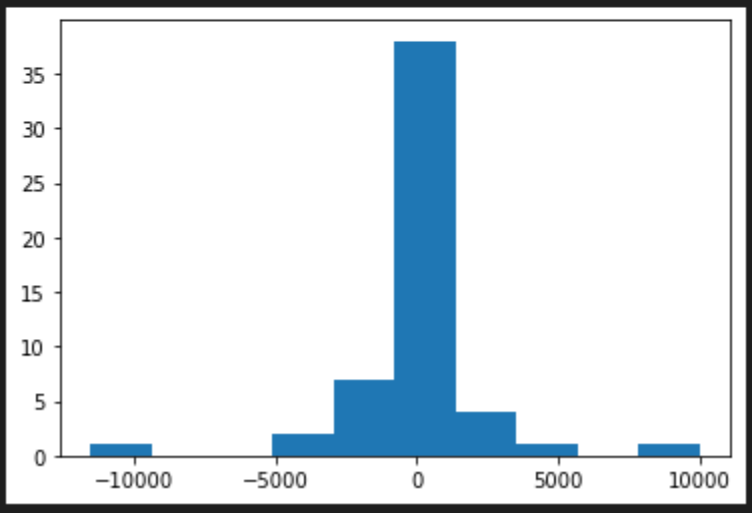
```
Jarque-Bera 310.24914664118234
Chi^2 two-tail prob. 4.268288823473115e-68
Skew -0.6702477064419257
Kurtosis 14.665829660437383
```

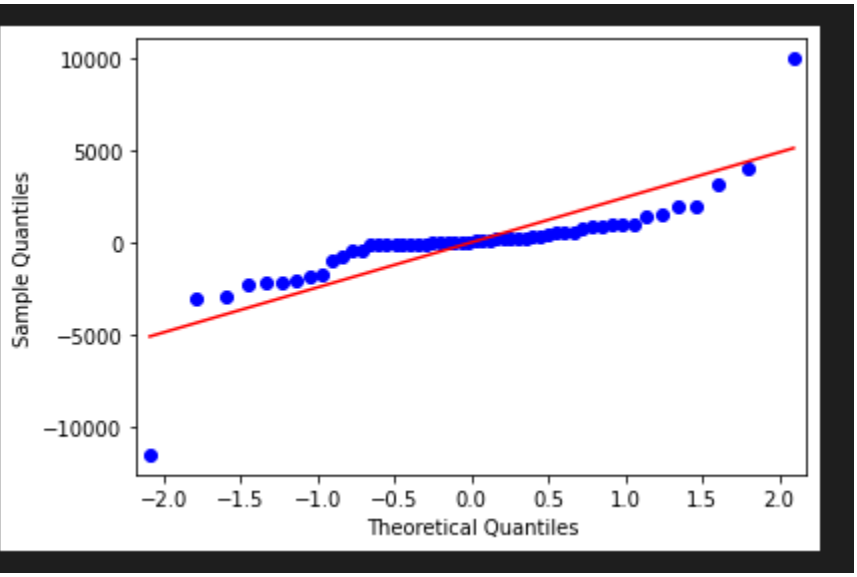
- Como nuestro Chi-cuadrado es: 4.268288823709131e-68 < 0.05, no podemos aceptar la hipótesis de la normalidad nula.
- El valor de nuestro Skew es: -0.6702477064416816, no podemos decir que es sigue un modelo simétrico ya que es no es próximo a 0.
- Nuestro valor de kurtosis es: 14.665829660435334, por lo que deberías rechazarlo ya que no es próximo a 4.

- Graficas:

```
from matplotlib import pyplot
from statsmodels.graphics.gofplots import qqplot
pyplot.hist(mco1.resid)
pyplot.show()
qqplot(mco1.resid, line='s')
pyplot.show()
```

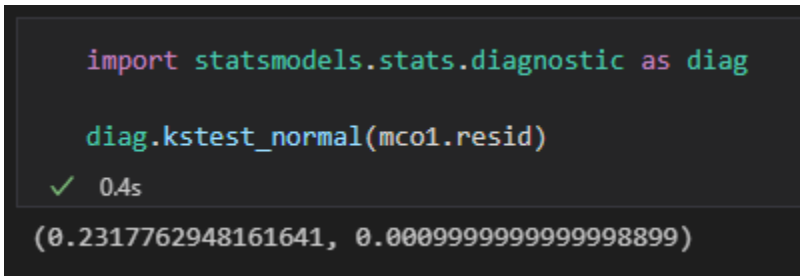
✓ 0.8s





- Como nuestros datos al menos la mitad de ellos se encuentran por encima de la media, no están del todo mal.

Kolmogorov-Smirnov: Test de Hipótesis para contrastar si una muestra proviene de una distribución (en este caso normal):



Como nuestro p-valor: 0.0009999999999998899 < 0.05, rechazamos la hipótesis.

Conclusiones

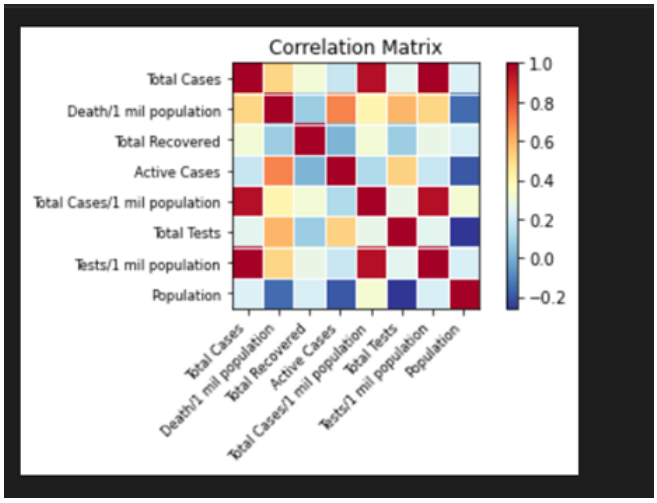
En primer lugar, una vez mostramos el primer modelo y vimos los resultados que dio, se podía observar que este modelo presentaba una serie de problemas:

- Linealidad: (No hay problema supuestamente)
  - Test de Harvey-Collier (0.41850016172762494 > 0.05)
  - Falla el test de Reset de Ramsey(0.0007690485400297567).

- Autocorrelación:
  - Durbin-Watson no tuvimos problemas ya que nuestro valor era próximo a 2 .

- Heterocedasticidad:
  - Test de goldfelquandt (0.337956702000184 > 0.05 no tenemos problema).
  - Test de Breush-Pagan (2.8404291550362514e-08 < 0.05 hay un problema de heterocedasticidad).
  - Test de White(3.642084309485573e-30 < 0.05 ya que es para muestras grandes).
  - Test de Glejser(“Total Tests”).

- Multicolinealidad:
  - Condition Number(72311132.56520036 > 900 las variables presentan alta dependencia):
  - Test de Vifs(Valores > 20, por lo que tenemos problema de multicolinealidad, en al menos 4 variables):
  - Matriz de correlaciones (Los datos presentan dependencia)



- Normalidad de los residuos
  - Jarque-Bera → nuestro p-valor es 5.192827366266701e-71, por lo que no podemos aceptar la hipótesis de normalidad nula, al ser menor que alfa.
  - Skew → nuestro valor no es próximo a 0 (es 1.5756829717965635), por lo que tampoco podemos decir que se sigue un modelo simétrico.
  - Kurtosis → nuestro valor es 14.572525607665003, lo cual no es aceptable, ya que no es próximo a 4.

Para los siguiente modelos, hemos seguido con esta estructura de comprobacion de hipotesis, y tras una larga serie de comprobaciones entre las distintas variable y situaciones que pudiese tener nuestro modelo, llegamos a la conclusión de que para obtener el modelo más fiable hasta el momento, era eliminando todos las variables que almacenarán datos totales, como por ejemplo, “Total Test” y quedarnos con aquellas que afectasen de una manera más porcentual a esta, como en nuestro caso son los “Test/1 mil population” por ejemplo.

Para el último modelo, hemos decidido quitar las variables que almacenarán datos totales que pudieran relacionarse con las porcentuales como previamente hemos explicado, por lo que nuestro resultado final sería el siguiente:

- Eliminamos → ["Total Cases", "Total Tests", "Active Cases",].
- Dejamos dentro del modelo → ["Total Recovered", "Death/1 mil population", "Total Cases/1 mil population", "Tests/1 mil population", "Population"].

Y vamos a ir viendo los distintos resultados y las posibles mejoras en el modelo:

- Linealidad: (No hay problema supuestamente)
  - ***Se produce una mejora en el test de harvey-collier(0.41850016172762494 > 0.950066315053199).***
  - Falla el test de Reset de Ramsey(0.0007690485400297567 < 0.05).

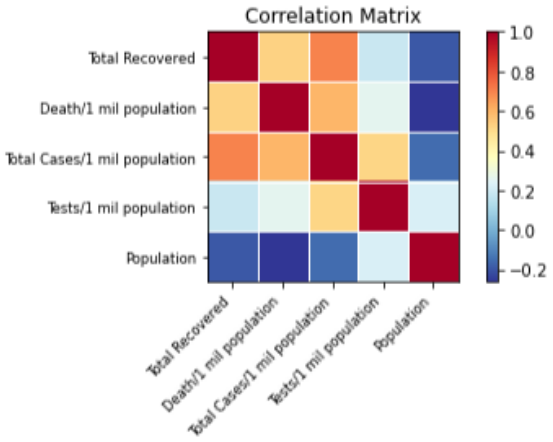
- Autocorrelación:
    - Durbin-Watson no tuvimos problemas ya que nuestro valor era próximo a 2.
        - Primer modelo (2.07389242837 tendencia positiva).
        - ***Este modelo (1.9549380214767726 una mejoría con tendencia negativa).***

- Heterocedasticidad:
    - Test de goldfelquandt (0.337956702000184 > 0.05 no tenemos problema).
    - ***Test de Breush-Pagan (0.07507412444391537 < 0.05 ya no hay un problema de heterocedasticidad).***
    - Test de White(3.642084309485573e-30 < 0.05 ya que es para muestras grandes, por lo tanto no se cumple).
    - Test de Glejser("Total Recovered", nos recomienda quitar esta variable, pero luego nos resultado no son muy favorables por lo que hemos decidido dejarlo como esta).

- Multicolinealidad:
    - ***Condition Number(72311132.56520036 > 70268054.63283534 se produce una reducción del condition number, lo que implica una menor relación entre las variables).***

    - ***Test de Vifs(Valores < 20, por lo que no tenemos problema de multicolinealidad y hemos conseguido mejorarlo respecto al anterior modelo).***

    - Matriz de correlaciones:
        - ***En este modelo podemos ver que ya no se presentan unas dependencias tan graves como anteriormente.***



- Normalidad de los residuos
  - Jarque-Bera → nuestro p-valor es 4.268288823709131e-68 > 5.192827366266701e-71, por lo que no podemos aceptar la hipótesis de normalidad nula, al ser menor que alfa, sin embargo hemos conseguido reducir dicho valor un poco.

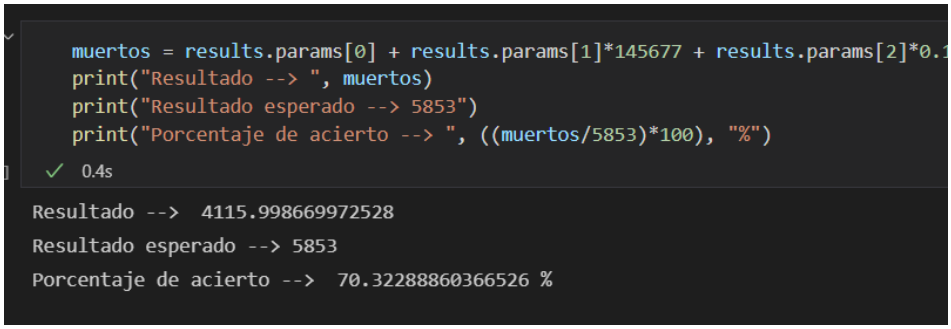
  - Skew → nuestro valor no es próximo a 0 (-0.6702477064416816), por lo que tampoco podemos decir que se sigue un modelo simétrico.
    - Anterior modelo 1.5756829717965635, aunque no se producía una simetría podemos observar que la mayoría de los valores se mantienen por encima de la línea media.
    - En este modelo, sin embargo, nuestro valor es de -0.6702477064416816, que implica que los valores son más simétricos, y además sus valores permanecen por debajo de la línea media.

  - Kurtosis → nuestro valor es 14.665829660435334, lo cual no es aceptable, ya que no es próximo a 4.
    - No se presentan diferencias significativas.

Predicciones sobre el modelo final

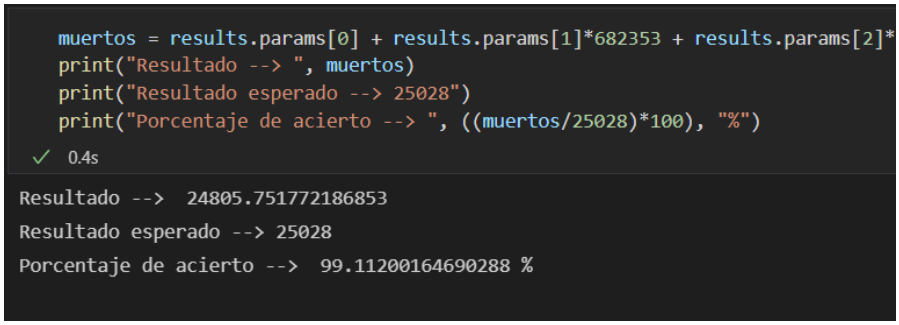
Algeria

"Total Recovered": 145677
"Death/1 mil population": 0.1364
"Total Cases/1 mil population": 4.7271
"Tests/1 mil population": 290
"Population": 44857594



Tunez

"Total Recovered": 682353
"Death/1 mil population": 2090
"Total Cases/1 mil population": 59271
"Tests/1 mil population": 250004
"Population": 11975999



Marruecos

"Total Recovered": 917807
"Death/1 mil population": 386
"Total Cases/1 mil population": 25067
"Tests/1 mil population": 262717
"Population": 37467214

```
    muertos = results.params[0] + results.params[1]*917807 + results.params[2]*386 +
    print("Resultado --> ", muertos)
    print("Resultado esperado --> 14457")
    print("Porcentaje de acierto --> ", ((muertos/14457)*100), "%")
5]  ✓ 0.4s
.  Resultado -->  25977.07783518474
   Resultado esperado --> 14457
   Porcentaje de acierto -->  179.6851202544424 %
```

- ***Este resultado no tiene sentido, hemos llegado a la conclusión de que se han reducido el número de muertes o que hay escasez de información respectos a nuestra predicción.***

Sudáfrica

"Total Recovered": 2791256
"Death/1 mil population": 1466
"Total Cases/1 mil population": 48326
"Tests/1 mil population": 298413
"Population": 11360709

```
    muertos = results.params[0] + results.params[1]*2791256 + results.params[2]*1466 + resul
    print("Resultado --> ", muertos)
    print("Resultado esperado --> 88317")
    print("Porcentaje de acierto --> ", ((muertos/88317)*100), "%")
3]  ✓ 0.2s
.  Resultado -->  84490.74274679863
   Resultado esperado --> 88317
   Porcentaje de acierto -->  95.66758692754354 %
```

Tabla de Resultados

Porcentaje de acierto	Muertos estimados	Muertos reales	Country
74	4309	5853	Algeria
77	1245	1622	Angola
56	283	159	Benin
93	2201	2376	Botswana
75	259	195	Burkina Faso
-703	-267	38	Burundi
-175	-605	346	Cabo Verde
78	1937	1517	Cameroon
92	92	100	CAR
14	24	174	Chad
25	591	147	Comoros
78	161	206	Congo
-488	-849	174	Djibouti
91	1191	1087	DRC
44	7726	17695	Egypt
-268	-413	154	Equatorial Guinea
39	108	42	Eritrea
55	2228	1227	Eswatini
66	9052	5990	Ethiopia
-1476	-2968	201	Gabon
92	368	339	Gambia
36	3197	1158	Ghana

70	549	383	Guinea
-30	-41	135	Guinea-Bissau
48	1391	666	Ivory Coast
75	6908	5181	Kenya
77	501	652	Lesotho
41	116	286	Liberia
62	7729	4775	Libya
85	1127	958	Madagascar
78	1783	2290	Malawi
47	260	553	Mali
95	744	785	Mauritania
-1781	-1888	106	Mauritius
56	25976	14457	Morocco
46	4147	1922	Mozambique
62	5690	3527	Namibia
18	36	204	Niger
55	5001	2747	Nigeria
-4	-52	1308	Rwanda
29	194	56	Sao Tome and Principe
93	2014	1868	Senegal
-735	-867	118	Seychelles
-25	-31	121	Sierra Leone
29	343	1180	Somalia
96	84344	88317	South Africa

90	145	130	South Sudan
33	965	2928	Sudan
-26	-189	723	Tanzania
87	270	236	Togo
99	24806	25028	Tunisia
82	2597	3176	Uganda
66	5530	3654	Zambia
84	3899	4637	Zimbabwe

En esta tabla, hemos obtenido las predicciones de todos los muertos que, según nuestro modelo, debería haber en cada país de África. En la primera columna podemos ver el porcentaje de acierto entre nuestro modelo y los muertos reales. Luego, en la segunda columna, se encuentran los muertos por COVID estimados por nuestro modelo. En la tercera columna, los muertos reales por COVID en cada país, y en la última columna el nombre de los países.

Conclusión sobre las predicciones:

Es muy complicado estimar datos sobre el covid, porque según nosotros y nuestro modelo, en Marruecos, por ejemplo, deberían de haber muerto por lo menos 27000 personas y no el resultado tan reducido que muestran en este país, creemos que es por la falta de información y de comunicación, buscándose números bajos para tranquilizar tanto a la población como al beneficio internacional.

Por lo que en general, en estos países no suelen darle la importancia que se merece a este virus, y sus datos no se asemejan a los de la realidad, por lo que aunque nuestro modelo pueda parecer consistente, no se va a poder verificar realmente si los resultados son fiables.

Sin embargo en países que en nuestra opinión creemos que están más desarrollados, como son Sudáfrica y Túnez, obtenemos unos datos bastante precisos, con un porcentaje de acierto de más del 90%, demostrando así la fiabilidad de nuestro modelo.

En la mayoría de países podemos ver que nuestro modelo consigue una tasa de acierto relativamente buena, acercándose y superando al 70%. Sin embargo, podemos entender que llame la atención que para algunos países, la cifra de muertos nos salga negativa, y nosotros, como explicación a esto hemos concluido que es debido a que los datos que obtuvimos de estos países no son para nada fiables, puesto que muchos de ellos no tienen ningún tipo de

sentido (hay países en los que se dicen que se hacen 2 test de COVID por cada millón de habitantes).