UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

POSGRADO EN INGENIERÍA DE SISTEMAS

CIENCIA DE DATOS

Portafolio Alberto Benavides

José Alberto Benavides Vázquez

1373079

Cd. Universitaria

3 de junio de 2019

Ciencia de datos

Práctica 2. Lectura y manipulación de datos con pandas

Alberto Benavides

Una vez extraídos los datos de los archivos PDF como se mencionó en la práctica anterior (https://nbviewer.jupyter.org/github/jbenavidesv87/CienciaDatos/blob/master/Notebooks/P1.ipynb), se lidió con los errores de lectura de las librerías y programas con los que se realizó el proceso. Los datos extraídos se guardaron en formato CSV y se ven como

```
2006|01|TOTAL| 'T�tanos\rCIE-10a REV.\rA34| '-'
2006|2006_sem23|TOTAL| 'Hepatitis\raguda tipo A\rCIE-10a REV.\rB15'| '276'
2006|2006_sem30|TOTAL| 'Varicela\rCIE-10a REV.\rB01'| '3 320'
```

Se optó por separar las columnas de los datos con `|` en lugar de `,`, separador por defecto, puesto que en algunos casos la separación por miles se denota con `,` o se utiliza este mismo símbolo para separar nombres de enfermedades. Por otro lado, la segunda columna que corresponde al número de la semana podía contener prefijos del nombre del archivo, como en `2006_sem23` asociado a la semana 23 del año 2006. Aunado a esto, la cuarta columna contiene tanto el nombre de la enfermedad como la CIE correspondiente y suele contener la revisión de la CIE con base en la cual se hizo la clasificación para cada enfermedad. Esto tiende a aparecer como `CIE-10a REV.`. También es posible encontrar símbolos para saltar líneas, espacios o tabulaciones como `\r`, `\n`, etc. Finalmente, es común que el algoritmo de extracción de datos desde tablas en archivos PDF incluya espacios en blanco al inicio o al final de los datos extraídos además de `'` o `"` cuando se toma la información de la celda como textual y no numérica.

Para solucionar todas estas eventualidades, primero se leyeron los archivos generados con `csv.reader` (https://docs.python.org/3/library/csv.html) especificando que `|` es el separador de columnas con

```
with open('test/{}enf.csv'.format(year), "r", newline='') as f:
    reader = csv.reader(f, delimiter='|')
```

La información obtenida se almacena en `reader` y, para poder manipular las columnas, se convierten a arreglos con

```
lines = list(reader)
```

De esta manera, se puede iterar entre registros

```
for i in range(len(lines)):
```

tal que, por ejemplo, `lines[i][1]` corresponde al registro `i`, segunda columna (los índices de columnas empiezan en `0`).

Ahora se sobreescriben los datos indeseables con los de interés. Primero se extrajo la semana con

```
lines[i][1] = lines[i][1][-2:]
```

El índice `[-2:]` se utiliza para tomar desde el penúltimo caracter hasta el último de la columna de la semana. Con ello se elimina todo lo correspondiente a los nombres de archivo y se conserva sólo el número de la semana, desde la `01` hasta la `53`. Por su parte, es necesario forzar la conversión de casos reportados (`lines[i][4]`) a cadenas de texto para poderlas manipular, lo que se logra con

```
lines[i][4] = str(lines[i][4])
```

En tanto los casos son cadenas de texto, es posible eliminar las `'` con

```
lines[i][4] = lines[i][4].replace("'", "")
```

que utiliza `string.replace` (https://docs.python.org/3.3/library/stdtypes.html#str.replace) para sustituir el primer argumento de la función por el segundo en la cadena de texto desde la que se llama el método. Ya que se eliminan las `'` se requiere eliminar espacios iniciales en la cadena mediante

```
lines[i][4] = re.sub("^\s+", "", lines[i][4])
```

Después, es necesario convertir los `-` a ceros

```
if lines[i][4][0] is "-":
  lines[i][4] = "0"
```

y los `n.e` o `n.d` a `NA`

```
lines[i][4] = lines[i][4].replace("n.e", "NA")
lines[i][4] = lines[i][4].replace("n.d", "NA")
```

El trato que se da a la columna de las enfermedades es similar. Primer se eliminan los espacios iniciales y finales

```
lines[i][3] = re.sub("\s+", " ", lines[i][3])
lines[i][3] = re.sub("^\s+", "", lines[i][3])
```

luego se eliminan los `'`

```
lines[i][3] = lines[i][3].replace("'", "")
```

se cambian los `\r` por espacios en blanco puesto que los saltos de línea en los cuadros de los PDFs corresponden con espacios en los nombres de las enfermedades

```
lines[i][3] = lines[i][3].replace("\\r", " ")
```

y, por estandarizar el nombre de las enfermedades, se convierten todas a minúsculas

```
lines[i][3] = str.lower(lines[i][3]) # a minúsculas
```

por último, cuando se encuentra `cie` en la columna correspondiente al nombre de la enfermedad, se almacenan los caracteres posteriores a `cie-10a rev.` en `cie` , variable temporal que se agregará como columna a la línea con la instrucción `append` (https://www.tutorialspoint.com/python3/list_append.htm)

```
if "cie" in lines[i][3]:
  cie = lines[i][3][lines[i][3].find("cie-10a rev.") + 13:]
  lines[i][3] = lines[i][3][:lines[i][3].find("cie-10a rev.") - 1]
  lines[i].append(cie)
```

Tras corregir todos estos detalles, se procede al guardado de estos archivos en CSV con `csv.writer` (https://docs.python.org/3/library/csv.html#csv.writer)

```
with open('csvSemanales/{}enf_limpio.csv'.format(year), "w", newline = "") as f:
  writer = csv.writer(f)
  writer.writerows(lines)
```

Ya que se han guardado los datos limpios, se procede a leerlos en R (https://www.r-project.org/) con

```
enfermedades <- read.csv(file="csvSemanales/enf.csv", header=TRUE, sep=",", stringsAsFactors=F
ALSE,
colClasses=c("integer", "integer", "character", "character", "integer", "character"))
```

en donde se especifica en el atributo `colClasses` el tipo de dato en que deben leerse las columnas. Enseguida se filtran los registros que en la columna `estado` contienen `TOTAL` , lo cual corresponde a la suma de todos los casos registrados en México durante la semana y el año especificado en el registro. Esto se realiza con `grep` (https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/grep)

```
enfermedades <- enfermedades[grep("TOTAL", enfermedades$estado),]
```

Tras hacer el preprocesamiento de datos descrito en la práctica 1, los datos obtenidos se agruparon en una columna con `cbind` (https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/cbind) para añadirse a un DataFrame (https://www.rdocumentation.org/packages/base/versions/3.5.2/topics/data.frame) con `rbind` (https://www.rdocumentation.org/packages/spectacles/versions/0.5-0/topics/rbind)

```
enfLimpias <- rbind(enfLimpias, cbind(coef(modeloLineal)[2], topFourier[1,1], topFourier[2,1],
 topFourier[3,1], topFourier[4,1], ac$acf[1], ac$acf[2], ac$acf[min(3, 4)], ac$acf[min(4, 8)],
 ac$acf[min(5, 16)], ac$acf[min(6, 32)], enfermedad$enfermedad, enfermedad$cie))
```

y se agregó el nombre de los encabezados mediante

```
colnames(enfLimpias) <- c("m", "f1", "f2", "f3", "f4", "ac1", "ac2", "ac3", "ac4", "ac5", "ac6
", "cie", "t")
```

para luego guardarse en un archivo CSV con

```
write.csv(enfLimpias, file = "semanalesTodas.csv",row.names=FALSE)
```

Estos datos fueron importados a un *dataframe* en Python mediante Pandas con

```
df = pd.read_csv('semanalesTodas.csv',sep=',')
```

Más adelante, StandartScaler (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html) se usa para estandarizar las características seleccionadas

```
features = ["m", "f1", "f2", "f3", "f4", "ac1", "ac2", "ac3", "ac4", "ac5", "ac6"]
x = df.loc[:, features].values
x = StandardScaler().fit_transform(x)
```

y se sobreescriben

```
df.loc[:, features] = x
```

Por último, para poder utilizar estos datos con la librería de NumPy (http://www.numpy.org/) es necesario extraer los valores

```
df = df.values
```

Ciencia de datos

Práctica 7. Regresión múltiple

Alberto Benavides

Ahora, se comprobarán las relaciones existentes entre las múltiples características y los agrupamientos generados por k-medias a partir de los datos con los que se ha trabajado en prácticas anteriores

In [1]:
```
import pandas as pd

data = pd.read_csv("D:/FIME/Epidemia/Data/semanalesTodasKmeans.csv")
print(data.sample(3))
```

```
                m        f1        f2        f3        f4        ac1       ac2  \
53 -3.855778e-05  0.004167  0.012500  0.016667  0.029167  0.741248  0.742676
98 -3.469620e-08  0.002083  0.004167  0.006250  0.437500  0.138975  0.213426
63  6.894726e-06  0.055556  0.018519  0.351852  0.222222  0.246428  0.201992

         ac4       ac8      ac16      ac32                       enf cie  \
53  0.586022  0.236282  0.147558 -0.029467          INFLUENZA A H1N1   J
98  0.143331  0.177340  0.217480  0.113430         SIFILIS CONGENITA   A
63  0.054887  0.107901  0.048868 -0.114340  TUMOR MALIGNO DE LA MAMA   C

    cluster
53        3
98        4
63        4
```

Primero se aplica una prueba de mínimos cuadrados ordinarios a cada característica (excluidas el nombre de la enfermedad y el grupo de CIE al que pertenecen) con respecto al resto de las características

In [48]:
```
import statsmodels.api as sm

features = data
features = features.loc[:, features.columns != 'enf']
features = features.loc[:, features.columns != 'cie']
for column in features:
    y = features[column]
    # https://stackoverflow.com/a/29763653
    x = features.loc[:, features.columns != column]
    x = sm.add_constant(x)
    m = sm.OLS(y, x).fit()
    print(m.summary())

import matplotlib.pyplot as plt
fig = plt.figure()
pd.plotting.scatter_matrix(features, figsize = (12, 12), s = 150)
plt.show()
plt.close(fig)

# Predicción
# https://www.statsmodels.org/devel/examples/notebooks/generated/ols.html
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      m   R-squared:                       0.090
Model:                            OLS   Adj. R-squared:                  0.004
```

```
Model:                            OLS   Adj. R-squared:                  0.004
Method:                 Least Squares   F-statistic:                     1.041
Date:                Fri, 22 Mar 2019   Prob (F-statistic):              0.416
Time:                        19:23:40   Log-Likelihood:                790.89
No. Observations:                 128   AIC:                            -1558.
Df Residuals:                     116   BIC:                            -1524.
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const       7.431e-05      0.000      0.309      0.758      -0.000       0.001
f1            -0.0004      0.001     -0.645      0.520      -0.002       0.001
f2            -0.0002      0.001     -0.481      0.632      -0.001       0.001
f3          8.145e-05      0.000      0.205      0.838      -0.001       0.001
f4             0.0003      0.000      0.717      0.475      -0.001       0.001
ac1        -8.912e-05      0.000     -0.222      0.825      -0.001       0.001
ac2           -0.0003      0.000     -0.703      0.484      -0.001       0.001
ac4        -2.971e-05      0.001     -0.055      0.956      -0.001       0.001
ac8         5.575e-05      0.000      0.137      0.892      -0.001       0.001
ac16           0.0007      0.000      1.560      0.122      -0.000       0.001
ac32          -0.0009      0.000     -2.119      0.036      -0.002    -5.9e-05
cluster    -6.226e-06    2.7e-05     -0.231      0.818   -5.97e-05    4.72e-05
==============================================================================
Omnibus:                      226.205   Durbin-Watson:                   2.107
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            30562.466
Skew:                          -7.666   Prob(JB):                         0.00
Kurtosis:                      77.131   Cond. No.                         57.2
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
                            OLS Regression Results
==============================================================================
Dep. Variable:                     f1   R-squared:                       0.590
Model:                            OLS   Adj. R-squared:                  0.552
Method:                 Least Squares   F-statistic:                     15.21
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           6.49e-18
Time:                        19:23:40   Log-Likelihood:                140.18
No. Observations:                 128   AIC:                            -256.4
Df Residuals:                     116   BIC:                            -222.1
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2437      0.032      7.736      0.000       0.181       0.306
m             -9.6396     14.956     -0.645      0.520     -39.261      19.982
f2            -0.0609      0.082     -0.740      0.461      -0.224       0.102
f3            -0.0208      0.064     -0.324      0.746      -0.148       0.106
f4            -0.1349      0.065     -2.072      0.041      -0.264      -0.006
ac1           -0.4392      0.050     -8.709      0.000      -0.539      -0.339
ac2            0.2492      0.068      3.655      0.000       0.114       0.384
ac4            0.0366      0.087      0.422      0.674      -0.135       0.208
ac8           -0.0584      0.066     -0.889      0.376      -0.188       0.072
ac16          -0.0242      0.069     -0.352      0.725      -0.160       0.112
ac32          -0.0137      0.070     -0.195      0.845      -0.152       0.125
cluster       -0.0156      0.004     -3.783      0.000      -0.024      -0.007
==============================================================================
Omnibus:                       15.155   Durbin-Watson:                   2.276
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               19.963
Skew:                           0.660   Prob(JB):                     4.63e-05
Kurtosis:                       4.415   Cond. No.                     7.25e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.25e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
Dep. Variable:                     f2   R-squared:                       0.546
```

```
Model:                            OLS   Adj. R-squared:                  0.503
Method:                 Least Squares   F-statistic:                     12.67
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           1.88e-15
Time:                        19:23:40   Log-Likelihood:                124.92
No. Observations:                 128   AIC:                            -225.8
Df Residuals:                     116   BIC:                            -191.6
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2104      0.039      5.383      0.000       0.133       0.288
m             -8.1037     16.862     -0.481      0.632     -41.501      25.293
f1            -0.0772      0.104     -0.740      0.461      -0.284       0.129
f3             0.0573      0.072      0.796      0.428      -0.085       0.200
f4             0.1100      0.074      1.487      0.140      -0.037       0.257
ac1           -0.3092      0.067     -4.602      0.000      -0.442      -0.176
ac2           -0.0732      0.081     -0.906      0.367      -0.233       0.087
ac4            0.2659      0.095      2.810      0.006       0.078       0.453
ac8           -0.0513      0.074     -0.692      0.490      -0.198       0.095
ac16          -0.1243      0.076     -1.625      0.107      -0.276       0.027
ac32           0.0858      0.078      1.095      0.276      -0.069       0.241

cluster       -0.0093      0.005     -1.934      0.056      -0.019       0.000
==============================================================================
Omnibus:                        3.440   Durbin-Watson:                   1.515
Prob(Omnibus):                  0.179   Jarque-Bera (JB):                3.440
Skew:                           0.392   Prob(JB):                        0.179
Kurtosis:                       2.823   Cond. No.                     7.25e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.25e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

OLS Regression Results

```
==============================================================================
Dep. Variable:                     f3   R-squared:                       0.408
Model:                            OLS   Adj. R-squared:                  0.352
Method:                 Least Squares   F-statistic:                     7.276
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           2.44e-09
Time:                        19:23:40   Log-Likelihood:                 92.668
No. Observations:                 128   AIC:                            -161.3
Df Residuals:                     116   BIC:                            -127.1
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.2134      0.053      4.055      0.000       0.109       0.318
m              4.4559     21.712      0.205      0.838     -38.548      47.460
f1            -0.0436      0.135     -0.324      0.746      -0.310       0.223
f2             0.0948      0.119      0.796      0.428      -0.141       0.331
f4             0.1125      0.096      1.178      0.241      -0.077       0.302
ac1           -0.2891      0.090     -3.209      0.002      -0.468      -0.111
ac2            0.1404      0.104      1.356      0.178      -0.065       0.346
ac4            0.1365      0.125      1.090      0.278      -0.112       0.384
ac8           -0.0660      0.095     -0.693      0.490      -0.255       0.123
ac16          -0.0221      0.099     -0.222      0.825      -0.219       0.175
ac32          -0.0375      0.101     -0.370      0.712      -0.238       0.163
cluster       -0.0206      0.006     -3.417      0.001      -0.032      -0.009
==============================================================================
Omnibus:                       15.261   Durbin-Watson:                   2.321
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               16.988
Skew:                           0.793   Prob(JB):                     0.000205
Kurtosis:                       3.820   Cond. No.                     7.26e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

OLS Regression Results

Dep. Variable:	f4	R-squared:	0.434
Model:	OLS	Adj. R-squared:	0.380
Method:	Least Squares	F-statistic:	8.077
Date:	Fri, 22 Mar 2019	Prob (F-statistic):	2.48e-10
Time:	19:23:40	Log-Likelihood:	97.097
No. Observations:	128	AIC:	-170.2
Df Residuals:	116	BIC:	-136.0
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.2629	0.049	5.418	0.000	0.167	0.359
m	14.9979	20.931	0.717	0.475	-26.459	56.455
f1	-0.2645	0.128	-2.072	0.041	-0.517	-0.012
f2	0.1699	0.114	1.487	0.140	-0.056	0.396
f3	0.1050	0.089	1.178	0.241	-0.072	0.282
ac1	-0.2637	0.087	-3.016	0.003	-0.437	-0.091
ac2	0.2588	0.098	2.643	0.009	0.065	0.453
ac4	-0.0838	0.121	-0.691	0.491	-0.324	0.156
ac8	-0.1381	0.091	-1.512	0.133	-0.319	0.043
ac16	-0.0331	0.096	-0.345	0.731	-0.223	0.157
ac32	-0.0527	0.098	-0.539	0.591	-0.246	0.141
cluster	-0.0148	0.006	-2.495	0.014	-0.027	-0.003

Omnibus:	1.420	Durbin-Watson:	2.011
Prob(Omnibus):	0.492	Jarque-Bera (JB):	1.152
Skew:	0.230	Prob(JB):	0.562
Kurtosis:	3.064	Cond. No.	7.24e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

OLS Regression Results

Dep. Variable:	ac1	R-squared:	0.886
Model:	OLS	Adj. R-squared:	0.875
Method:	Least Squares	F-statistic:	81.86
Date:	Fri, 22 Mar 2019	Prob (F-statistic):	2.52e-49
Time:	19:23:40	Log-Likelihood:	94.256
No. Observations:	128	AIC:	-164.5
Df Residuals:	116	BIC:	-130.3
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.4626	0.035	13.145	0.000	0.393	0.532
m	-4.7558	21.444	-0.222	0.825	-47.228	37.716
f1	-0.9001	0.103	-8.709	0.000	-1.105	-0.695
f2	-0.4993	0.108	-4.602	0.000	-0.714	-0.284
f3	-0.2820	0.088	-3.209	0.002	-0.456	-0.108
f4	-0.2757	0.091	-3.016	0.003	-0.457	-0.095
ac2	0.4860	0.093	5.243	0.000	0.302	0.670
ac4	0.2739	0.122	2.252	0.026	0.033	0.515
ac8	-0.0689	0.094	-0.733	0.465	-0.255	0.117
ac16	-0.2554	0.095	-2.678	0.008	-0.444	-0.066
ac32	0.0393	0.100	0.393	0.695	-0.159	0.237
cluster	-0.0254	0.006	-4.392	0.000	-0.037	-0.014

Omnibus:	2.483	Durbin-Watson:	2.062
Prob(Omnibus):	0.289	Jarque-Bera (JB):	2.063
Skew:	-0.301	Prob(JB):	0.357
Kurtosis:	3.155	Cond. No.	7.22e+03

Warnings:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.22e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

OLS Regression Results

Dep. Variable:	ac2	R-squared:	0.876
Model:	OLS	Adj. R-squared:	0.864
Method:	Least Squares	F-statistic:	74.56
Date:	Fri, 22 Mar 2019	Prob (F-statistic):	2.83e-47
Time:	19:23:40	Log-Likelihood:	107.65
No. Observations:	128	AIC:	-191.3
Df Residuals:	116	BIC:	-157.1
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.1189	0.049	-2.437	0.016	-0.215	-0.022
m	-13.5478	19.277	-0.703	0.484	-51.728	24.632
f1	0.4143	0.113	3.655	0.000	0.190	0.639
f2	-0.0959	0.106	-0.906	0.367	-0.306	0.114
f3	0.1111	0.082	1.356	0.178	-0.051	0.273
f4	0.2195	0.083	2.643	0.009	0.055	0.384
ac1	0.3942	0.075	5.243	0.000	0.245	0.543
ac4	0.7067	0.091	7.790	0.000	0.527	0.886
ac8	-0.0490	0.085	-0.578	0.564	-0.217	0.119
ac16	0.1340	0.088	1.529	0.129	-0.040	0.308
ac32	-0.0953	0.090	-1.063	0.290	-0.273	0.082
cluster	0.0199	0.005	3.754	0.000	0.009	0.030

Omnibus:	2.278	Durbin-Watson:	2.136
Prob(Omnibus):	0.320	Jarque-Bera (JB):	1.969
Skew:	0.093	Prob(JB):	0.374
Kurtosis:	3.578	Cond. No.	7.21e+03

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.21e+03. This might indicate that there are
strong multicollinearity or other numerical problems.

OLS Regression Results

Dep. Variable:	ac4	R-squared:	0.899
Model:	OLS	Adj. R-squared:	0.890
Method:	Least Squares	F-statistic:	94.14
Date:	Fri, 22 Mar 2019	Prob (F-statistic):	1.93e-52
Time:	19:23:40	Log-Likelihood:	131.60
No. Observations:	128	AIC:	-239.2
Df Residuals:	116	BIC:	-205.0
Df Model:	11		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-0.0532	0.041	-1.291	0.199	-0.135	0.028
m	-0.8847	16.021	-0.055	0.956	-32.615	30.846
f1	0.0418	0.099	0.422	0.674	-0.155	0.238
f2	0.2396	0.085	2.810	0.006	0.071	0.408
f3	0.0743	0.068	1.090	0.278	-0.061	0.209
f4	-0.0489	0.071	-0.691	0.491	-0.189	0.091
ac1	0.1528	0.068	2.252	0.026	0.018	0.287
ac2	0.4861	0.062	7.790	0.000	0.362	0.610
ac8	0.4316	0.058	7.454	0.000	0.317	0.546
ac16	-0.0188	0.073	-0.256	0.799	-0.164	0.127
ac32	0.0483	0.075	0.648	0.518	-0.099	0.196
cluster	-0.0008	0.005	-0.167	0.868	-0.010	0.008

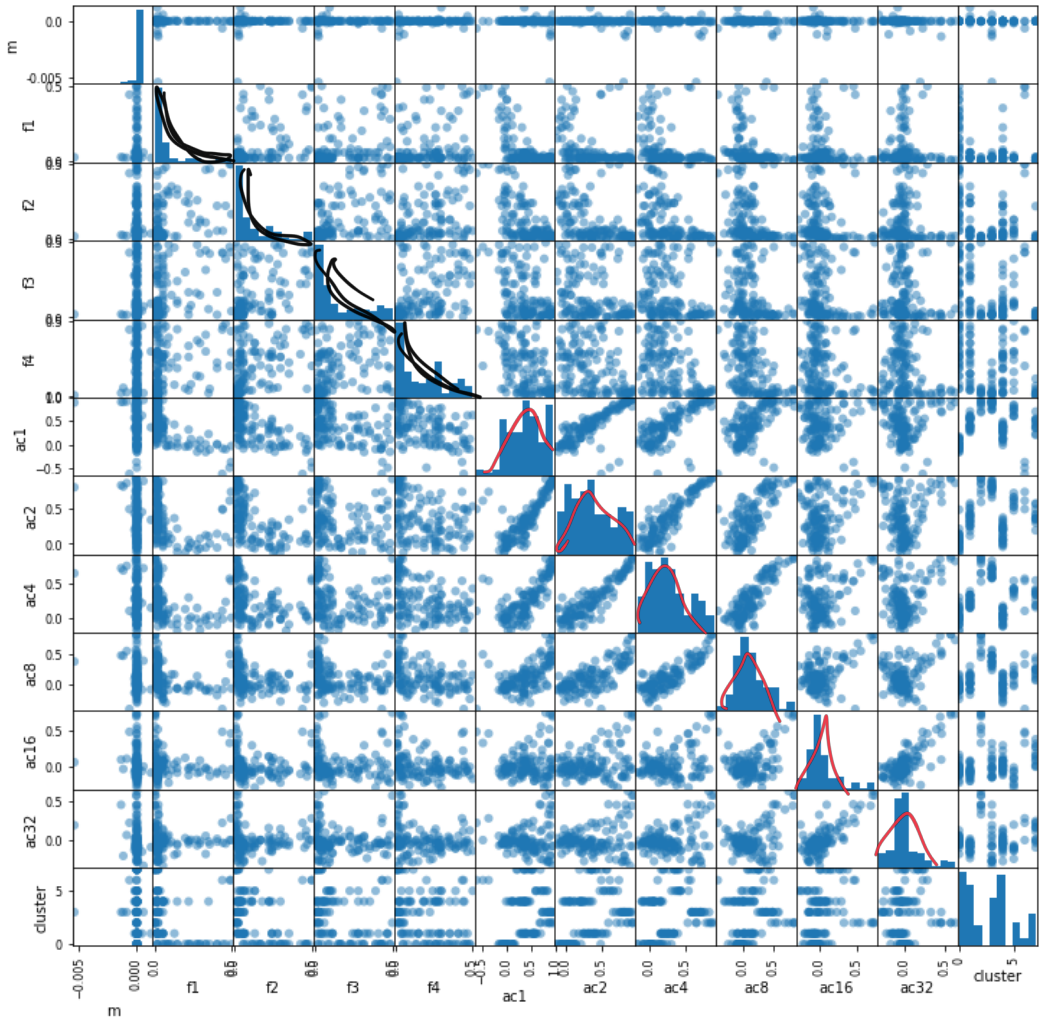
Omnibus:	16.412	Durbin-Watson:	1.978
Prob(Omnibus):	0.000	Jarque-Bera (JB):	42.584
Skew:	-0.405	Prob(JB):	5.66e-10
Kurtosis:	5.707	Cond. No.	7.24e+03

```
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.24e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
Dep. Variable:                    ac8   R-squared:                       0.778
Model:                            OLS   Adj. R-squared:                  0.756
Method:                 Least Squares   F-statistic:                     36.86
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           9.26e-33
Time:                        19:23:40   Log-Likelihood:                 96.198
No. Observations:                 128   AIC:                            -168.4
Df Residuals:                     116   BIC:                            -134.2
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0385      0.055      0.706      0.482      -0.070       0.147
m              2.8863     21.124      0.137      0.892     -38.952      44.725

f1            -0.1160      0.130     -0.889      0.376      -0.374       0.142
f2            -0.0803      0.116     -0.692      0.490      -0.310       0.149
f3            -0.0624      0.090     -0.693      0.490      -0.241       0.116
f4            -0.1400      0.093     -1.512      0.133      -0.323       0.043
ac1           -0.0669      0.091     -0.733      0.465      -0.248       0.114
ac2           -0.0586      0.101     -0.578      0.564      -0.259       0.142
ac4            0.7504      0.101      7.454      0.000       0.551       0.950
ac16           0.2378      0.094      2.523      0.013       0.051       0.424
ac32           0.0104      0.099      0.105      0.916      -0.185       0.205
cluster       -0.0077      0.006     -1.270      0.207      -0.020       0.004
==============================================================================
Omnibus:                        8.595   Durbin-Watson:                   2.068
Prob(Omnibus):                  0.014   Jarque-Bera (JB):               12.881
Skew:                          -0.310   Prob(JB):                      0.00160
Kurtosis:                       4.425   Cond. No.                     7.26e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
Dep. Variable:                   ac16   R-squared:                       0.693
Model:                            OLS   Adj. R-squared:                  0.664
Method:                 Least Squares   F-statistic:                     23.83
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           6.92e-25
Time:                        19:23:40   Log-Likelihood:                 101.51
No. Observations:                 128   AIC:                            -179.0
Df Residuals:                     116   BIC:                            -144.8
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0695      0.052      1.334      0.185      -0.034       0.173
m             31.2827     20.057      1.560      0.122      -8.443      71.008
f1            -0.0442      0.126     -0.352      0.725      -0.293       0.204
f2            -0.1792      0.110     -1.625      0.107      -0.397       0.039
f3            -0.0192      0.087     -0.222      0.825      -0.191       0.152
f4            -0.0309      0.090     -0.345      0.731      -0.208       0.147
ac1           -0.2280      0.085     -2.678      0.008      -0.397      -0.059
ac2            0.1474      0.096      1.529      0.129      -0.044       0.338
ac4           -0.0300      0.117     -0.256      0.799      -0.263       0.203
ac8            0.2188      0.087      2.523      0.013       0.047       0.391
ac32           0.7280      0.066     11.018      0.000       0.597       0.859
cluster        0.0064      0.006      1.099      0.274      -0.005       0.018
==============================================================================
Omnibus:                        4.684   Durbin-Watson:                   2.468
```

```
Prob(Omnibus):                  0.096   Jarque-Bera (JB):                 5.188
Skew:                          -0.218   Prob(JB):                        0.0747
Kurtosis:                       3.885   Cond. No.                      7.19e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.19e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
Dep. Variable:                   ac32   R-squared:                       0.635
Model:                            OLS   Adj. R-squared:                  0.601
Method:                 Least Squares   F-statistic:                     18.37
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           1.08e-20
Time:                        19:23:40   Log-Likelihood:                 103.79
No. Observations:                 128   AIC:                            -183.6
Df Residuals:                     116   BIC:                            -149.4
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0091      0.052     -0.178      0.859      -0.111       0.093
m            -41.3989     19.533     -2.119      0.036     -80.087      -2.710
f1            -0.0241      0.123     -0.195      0.845      -0.268       0.220
f2             0.1193      0.109      1.095      0.276      -0.096       0.335
f3            -0.0315      0.085     -0.370      0.712      -0.200       0.137
f4            -0.0475      0.088     -0.539      0.591      -0.222       0.127
ac1            0.0339      0.086      0.393      0.695      -0.137       0.204
ac2           -0.1012      0.095     -1.063      0.290      -0.290       0.087
ac4            0.0746      0.115      0.648      0.518      -0.154       0.303
ac8            0.0092      0.087      0.105      0.916      -0.164       0.182
ac16           0.7024      0.064     11.018      0.000       0.576       0.829
cluster       -0.0048      0.006     -0.837      0.404      -0.016       0.007
==============================================================================
Omnibus:                       15.995   Durbin-Watson:                   2.074
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               32.401
Skew:                           0.500   Prob(JB):                     9.21e-08
Kurtosis:                       5.253   Cond. No.                      7.12e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.12e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
                            OLS Regression Results
==============================================================================
Dep. Variable:                 cluster   R-squared:                       0.390
Model:                             OLS   Adj. R-squared:                  0.332
Method:                 Least Squares   F-statistic:                     6.734
Date:                Fri, 22 Mar 2019   Prob (F-statistic):           1.20e-08
Time:                        19:23:40   Log-Likelihood:                -251.39
No. Observations:                 128   AIC:                             526.8
Df Residuals:                     116   BIC:                             561.0
Df Model:                          11
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          5.5121      0.649      8.493      0.000       4.227       6.798
m            -73.6204    319.190     -0.231      0.818    -705.816     558.575
f1            -7.0625      1.867     -3.783      0.000     -10.760      -3.365
f2            -3.3432      1.729     -1.934      0.056      -6.767       0.080
f3            -4.4464      1.301     -3.417      0.001      -7.023      -1.869
f4            -3.4343      1.377     -2.495      0.014      -6.161      -0.708
ac1           -5.6202      1.280     -4.392      0.000      -8.155      -3.086
ac2            5.4395      1.449      3.754      0.000       2.570       8.309
ac4           -0.3091      1.850     -0.167      0.868      -3.973       3.355
ac8           -1.7700      1.394     -1.270      0.207      -4.530       0.990
ac16           1.5994      1.455      1.099      0.274      -1.283       4.481
```

```
ac32           -1.2426      1.485     -0.837      0.404      -4.183       1.698
==============================================================================
Omnibus:                       13.026   Durbin-Watson:                   1.938
Prob(Omnibus):                  0.001   Jarque-Bera (JB):                9.275
Skew:                           0.536   Prob(JB):                      0.00968
Kurtosis:                       2.231   Cond. No.                     2.47e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.47e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

```
<Figure size 432x288 with 0 Axes>
```



Esta prueba muestra que las autocorrelaciones con retraso de 4 semanas (un mes) presentan el mejor ajuste R^2 respecto al resto de las características, además que del conjunto de las características resultan (significativas)[https://www.investopedia.com/terms/t/t-test.asp (https://www.investopedia.com/terms/t/t-test.asp)] las autocorrelaciones con retraso de 1, 2 y 8 semanas y el segundo coeficiente de Fourier con $P(t) < 0.05$. Pese a ello, el conjunto de características resultan significativas con respecto al (estadístico F)[http://efavdb.com/interpret-linear-regression/ (http://efavdb.com/interpret-linear-regression/)]. Todo esto parece indicar que se pueden predecir las autocorrelaciones mensuales principalmente a partir de las características mencionadas.

En cuanto a la predicción del grupo principal de la CIE, se tiene

In [25]:
```
import numpy as np

classnames, indices = np.unique(data.cie, return_inverse=True)
y = indices
# https://stackoverflow.com/a/29763653
x = features
x = sm.add_constant(x)
m = sm.OLS(y, x).fit()
print(m.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.161
Model:                            OLS   Adj. R-squared:                  0.073
Method:                 Least Squares   F-statistic:                     1.837
Date:                Fri, 22 Mar 2019   Prob (F-statistic):             0.0500
Time:                        19:01:17   Log-Likelihood:                -410.04
No. Observations:                 128   AIC:                             846.1
Df Residuals:                     115   BIC:                             883.2
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.5899      2.867     -0.903      0.368      -8.269       3.089
m            960.3029   1107.406      0.867      0.388   -1233.256    3153.862
f1            -0.1330      6.863     -0.019      0.985     -13.727      13.461
f2            11.5012      6.092      1.888      0.062      -0.565      23.568
f3             8.4751      4.735      1.790      0.076      -0.903      17.854
f4             3.4690      4.901      0.708      0.481      -6.240      13.178
ac1            6.6749      4.794      1.392      0.166      -2.821      16.171
ac2           10.4405      5.323      1.962      0.052      -0.103      20.984
ac4           -8.0055      6.418     -1.247      0.215     -20.718       4.707
ac8           -1.9742      4.867     -0.406      0.686     -11.615       7.667
ac16           2.6113      5.073      0.515      0.608      -7.438      12.661
ac32          -4.3059      5.165     -0.834      0.406     -14.536       5.924
cluster        0.2886      0.322      0.896      0.372      -0.349       0.927
==============================================================================
Omnibus:                       13.893   Durbin-Watson:                   2.130
Prob(Omnibus):                  0.001   Jarque-Bera (JB):               15.890
Skew:                           0.863   Prob(JB):                     0.000354
Kurtosis:                       3.003   Cond. No.                     7.26e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Con esto, se puede observar que las características seleccionadas no son estadísticamente significativas para asignar series de tiempo de enfermedades a un grupo de CIE específico. Sin embargo, y sólo por comprobar,

In [31]:
```
y = data.cluster
# https://stackoverflow.com/a/29763653
x = features
x = sm.add_constant(x)
m = sm.OLS(y, x).fit()
print(m.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                cluster   R-squared:                       1.000
Model:                            OLS   Adj. R-squared:                  1.000
Method:                 Least Squares   F-statistic:                 2.691e+31
Date:                Fri, 22 Mar 2019   Prob (F-statistic):               0.00
Time:                        19:11:57   Log-Likelihood:                 4204.0
No. Observations:                 128   AIC:                            -8382.
Df Residuals:                     115   BIC:                            -8345.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const      -3.261e-15   6.34e-16     -5.143      0.000   -4.52e-15   -2.01e-15
m          -1.506e-12   2.45e-13     -6.150      0.000   -1.99e-12   -1.02e-12
f1          4.163e-15   1.52e-15      2.743      0.007    1.16e-15    7.17e-15
f2          5.412e-15   1.35e-15      4.017      0.000    2.74e-15    8.08e-15
f3          1.211e-15   1.05e-15      1.156      0.250   -8.64e-16    3.29e-15
f4           2.72e-15   1.08e-15      2.509      0.014    5.73e-16    4.87e-15
ac1         2.581e-15   1.06e-15      2.434      0.016    4.81e-16    4.68e-15
ac2        -1.221e-15   1.18e-15     -1.037      0.302   -3.55e-15    1.11e-15
ac4        -2.054e-15   1.42e-15     -1.447      0.151   -4.87e-15    7.58e-16
ac8         3.331e-15   1.08e-15      3.094      0.002     1.2e-15    5.46e-15
ac16        2.637e-16   1.12e-15      0.235      0.815   -1.96e-15    2.49e-15
ac32       -3.386e-15   1.14e-15     -2.964      0.004   -5.65e-15   -1.12e-15
cluster         1.0000   7.12e-17    1.4e+16      0.000       1.000       1.000
==============================================================================
Omnibus:                       41.935   Durbin-Watson:                   2.046
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              166.473
Skew:                          -1.078   Prob(JB):                     7.09e-37
Kurtosis:                       8.154   Cond. No.                     7.26e+03
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 7.26e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```

resulta evidente para la definición de agrupamientos por k-medias, se tiene el mejor ajuste y resultan significativas la mayoría de las características con $P(t) < 0.05$, excepto el cuarto mayor coeficiente de Fourier, y las autocorrelaciones con retraso de 2 y 16 semanas.

Práctica 8

clustering atributos

Alberto Benavides

Anova

Se realiza una ANOVA para ver las interacciones entre los casos normalizados y las semanas con las CIEs

```
from statsmodels.formula.api import ols
                                + casos * sem
m = ols('cie ~ casos + sem', data = features).fit()
a = sm.stats.anova_lm(m, typ = 2)
print(a)
n = len(a)
alpha = 0.05
for i in range(n):
    print("{:s} {:s}es significativo".format(a.index[i], "" if a['PR(>F)'][i] < alpha else "no "))
```

```
                sum_sq       df          F    PR(>F)
casos     9.609452e+02      1.0  22.552909  0.000002
sem       6.349658e+02      1.0  14.902334  0.000114
Residual  1.001768e+06  23511.0        NaN       NaN
casos es significativo
sem es significativo
Residual no es significativo
```

Las interacciones son estadísticamente significativas, pero los residuales son muy grandes. También, la suma de cuadrados para los casos y las semanas es similar, por lo que sus varianzas son cercanas, lo que ya se había constatado al estudiar su correlación.

PCA

Ahora se realiza un análisis de componentes principales o PCA por sus siglas en inglés a partir de las características

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

x = features[['casos', 'sem']].values
x = StandardScaler().fit_transform(x)
y = features.loc[:,['cie']].values
k = 2 # dimensiones deseadas
pca = PCA(n_components = k)
cd = pd.DataFrame(data = pca.fit_transform(x), columns = ['pca{:d}'.format(i) for i in range(k)])
cd['cie'] = y
ordenado = pd.DataFrame.sort_values(cd, ['cie'], ascending = False)
cd.sample(3)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	pca0	pca1	cie
11759	7.427456	7.499567	8.0
15772	-0.670430	0.439099	18.0
4183	0.691771	-0.953808	0.0

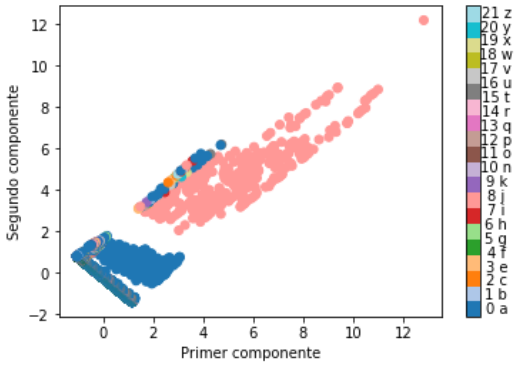
```
pca.explained_variance_ratio_
```

```
array([0.53307324, 0.46692676])
```

Estos componentes principales comparten de manera muy cercana la mitad de la varianza de los datos.

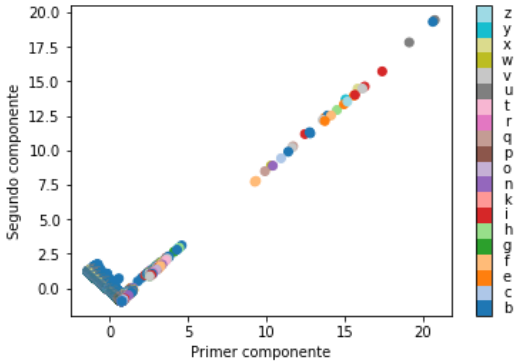
```
import matplotlib.cm as cm

plt.figure()
plt.xlabel('Primer componente')
plt.ylabel('Segundo componente')
plt.scatter(cd.pca0, cd.pca1, c = cd.cie, cmap=cm.tab20)
```

```
cbar = plt.colorbar()
cbar.ax.get_yaxis().set_ticks([])
for j, lab in enumerate(cieName):
    cbar.ax.text(2, (j + 0.5) / len(cieName), str(j) + ' ' +lab, ha='center', va='center')
cbar.ax.get_yaxis().labelpad = 15
plt.show()
```



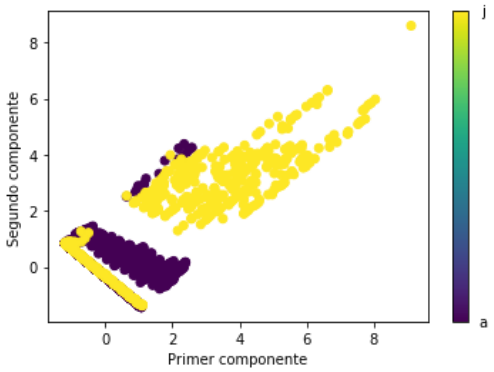
Estos resultados muestran una clara distinción entre las CIE A y J, por lo que podrían apartarse del conjunto y ver cómo se comportan el resto

```
features_copy = features

features_copy = features_copy[features_copy['cie'] != 0]
features_copy = features_copy[features_copy['cie'] != 8]

x = features_copy[['casos', 'sem']].values
x = StandardScaler().fit_transform(x)
y = features_copy.loc[:,['cie']].values
k = 2 # dimensiones deseadas
pca = PCA(n_components = k)
cd = pd.DataFrame(data = pca.fit_transform(x), columns = ['pca{:d}'.format(i) for i in range(k)])
cd['cie'] = y
ordenado = pd.DataFrame.sort_values(cd, ['cie'], ascending = False)

plt.figure()
plt.xlabel('Primer componente')
plt.ylabel('Segundo componente')
plt.scatter(cd.pca0, cd.pca1, c = cd.cie, cmap=cm.tab20)

cieName_copy = cieName
cieName_copy.remove('a')
cieName_copy.remove('j')

cbar = plt.colorbar()
cbar.ax.get_yaxis().set_ticks([])
for j, lab in enumerate(cieName_copy):
    cbar.ax.text(2, (j + 0.5) / len(cieName_copy), lab, ha='center', va='center')
cbar.ax.get_yaxis().labelpad = 15
plt.show()
```



Después de la eliminación, el resto de las CIE parecen seguir patrones impredecibles con base en estas características. En cambio, puede suponerse que entre las A y las J hay una clara distinción y probablemente se puedan obtener mejores modelos si se descartan las demás CIE.

```
features_copy = pd.concat([features[features.cie == 0], features[features.cie == 8]])

x = features_copy[['casos', 'sem']].values
x = StandardScaler().fit_transform(x)
y = features_copy.loc[:,['cie']].values
k = 2 # dimensiones deseadas
pca = PCA(n_components = k)
cd = pd.DataFrame(data = pca.fit_transform(x), columns = ['pca{:d}'.format(i) for i in range(k)])
cd['cie'] = y
ordenado = pd.DataFrame.sort_values(cd, ['cie'], ascending = False)

plt.figure()
plt.xlabel('Primer componente')
plt.ylabel('Segundo componente')
plt.scatter(cd.pca0, cd.pca1, c = cd.cie)

cieName_copy = ['a', 'j']

cbar = plt.colorbar()
cbar.ax.get_yaxis().set_ticks([])
for j, lab in enumerate(cieName_copy):
    cbar.ax.text(2, (j), lab, ha='center', va='center')
cbar.ax.get_yaxis().labelpad = 15
plt.show()
```




```
m = ols('cie ~ casos + sem', data = features_copy).fit()
```

```
a = sm.stats.anova_lm(m, typ = 2)
print(a)
n = len(a)
alpha = 0.05
for i in range(n):
    print("{:s} {:s}es significativo".format(a.index[i], "" if a['PR(>F)'][i] < alpha else "no "))
```

```
                sum_sq       df            F        PR(>F)
casos     14906.513429      1.0  3023.303363  0.000000e+00
sem         505.752055      1.0   102.575421  5.266110e-24
Residual  55315.710699  11219.0          NaN           NaN
casos es significativo
sem es significativo
Residual no es significativo
```

```
(1.001768e+06 - 55560.613600) / 1.001768e+06
```

```
0.9445374441986567
```

La suma cuadrada de los residuales se redujo un 94% pero aún así siguen siendo altos, por lo que con estas características parece improbable ajustar un modelo.

jbenavidesv87 Práctica 9 completa 22085d9 10 hours ago

1 contributor

3342 lines (2224 sloc) | 65.8 KB

Raw | Blame | History

Series de tiempo

De entre las series de tiempo para cada enfermedad en el periodo descrito, no todas contienen información de casos reportados durante el periodo de tiempo establecido para la investigación, por lo que se extraen los que reportan al menos la mitad del periodo (260 semanas)

```
# https://stackoverflow.com/a/16916611
print('Iniciales {}'.format(len(cie)))
cie = cie.filter(lambda x: x['sem'].count() >= 260 )
cie.reset_index(drop=True, inplace=True)
cie = cie.groupby('cie')
print('Restantes {}'.format(len(cie)))
```

```
Iniciales 138
Restantes 40
```

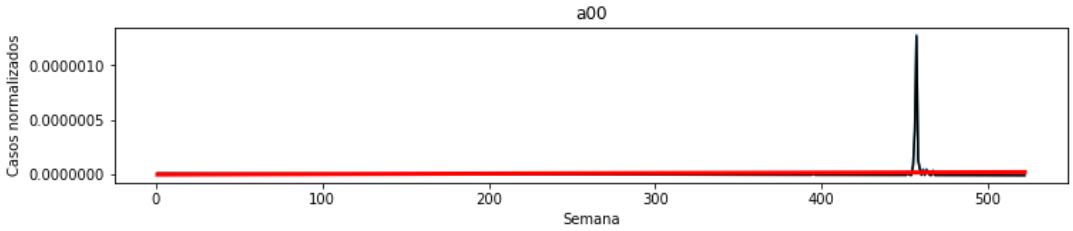
```
print('Iniciales {}'.format(len(cieG)))
cieG = cieG.filter(lambda x: x['sem'].count() >= 260 )
cieG.reset_index(drop=True, inplace=True)
cieG = cieG.groupby(cieG.cie.str[0])
print('Restantes {}'.format(len(cieG)))
```

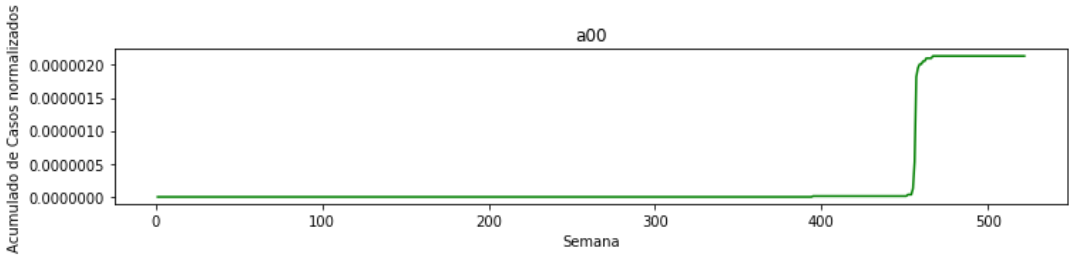
```
Iniciales 22
Restantes 12
```

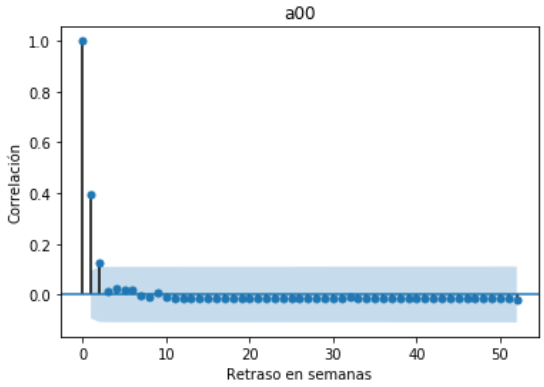
Así, de 138 series de tiempo de enfermedades, se obtienen 40 en las que al menos se cuenta con datos semanales de 5 años. Para dichas enfermedades se obtienen los pesos de la regresión lineal y se obtiene la serie de tiempo sin la tendencia y las autocorrelaciones (eliminando la a92.3 porque viene vacía)

```
from scipy import signal
from statsmodels.graphics.tsaplots import plot_pacf, plot_acf
from statsmodels.tsa.stattools import acf

ciesF = [] # CIEs Características
ciesTSt = [] # CIEs Series de tiempo

for name, group in cie:
    if name == 'a92.3':
        continue

    # https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.detrend.html
    detrended = signal.detrend(group.casos)

    a, b, r, p, e = stats.linregress(group['sem'], group.casos)
    print("y = f(x) = {} x + {}".format(a, b))
    print("error", e)
    print("p = ", p)
    print("pendiente {:s}significativa".format("no " if p >= 0.05 else ""))
    print("R^2", r**2)
    plt.figure(figsize=(12, 2))
    plt.plot(group['sem'], group.casos)
    plt.plot(group['sem'], detrended, c='black')
    plt.plot(group['sem'], (a * group['sem'] + b), label = 'y = {:.1f}x + {:.0f}'.format(a, b), color =
    plt.title(name)
    plt.xlabel("Semana")
    plt.ylabel("Casos normalizados")
    plt.show()

    # https://stackoverflow.com/questions/48497756/time-series-distance-metric
    plt.figure(figsize=(12, 2))
    # https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.cumsum.html
    plt.plot(group['sem'], group.casos.cumsum(), c='green')
    plt.title(name)
    plt.xlabel("Semana")
    plt.ylabel("Acumulado de Casos normalizados")
    plt.show()

    # https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/
    plot_acf(detrended, lags=52)
    # https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.acf.html
    plt.title(name)
    plt.xlabel("Retraso en semanas")
    plt.ylabel('Correlación')
    plt.show()

    temp = [a, b]
    # https://stackoverflow.com/a/3748071
    temp.extend(acf(detrended, nlags=52))
    temp.append(name)
    ciesF.append(temp)

    temp2 = list(group.casos)
    #temp2.append(name)
    ciesTSt.append(temp2)
```

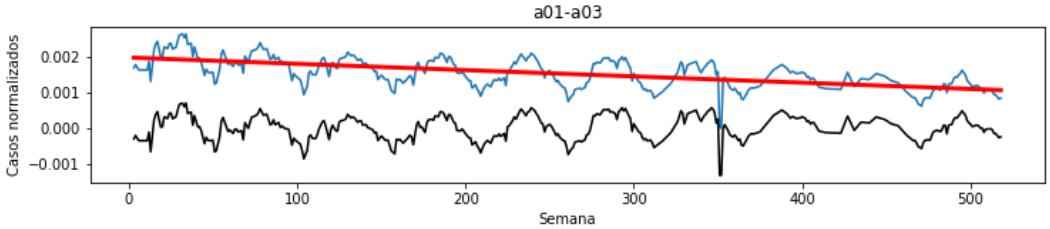
```
y = f(x) = 4.2687498879775535e-11 x + -5.61276490864389e-09
error 2.0113605583063735e-11
p =  0.03438325581830106
pendiente significativa
R^2 0.010390302941758996
```

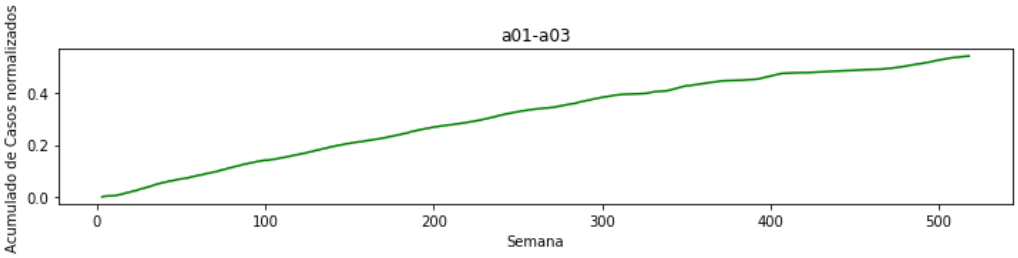


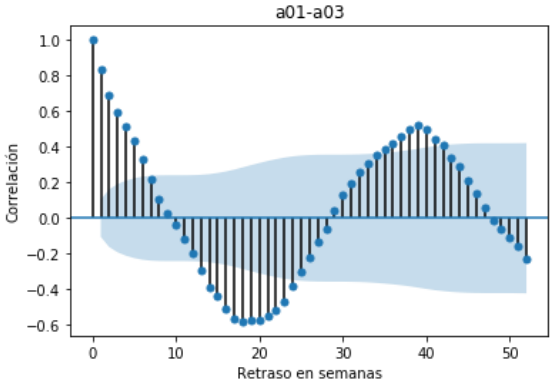




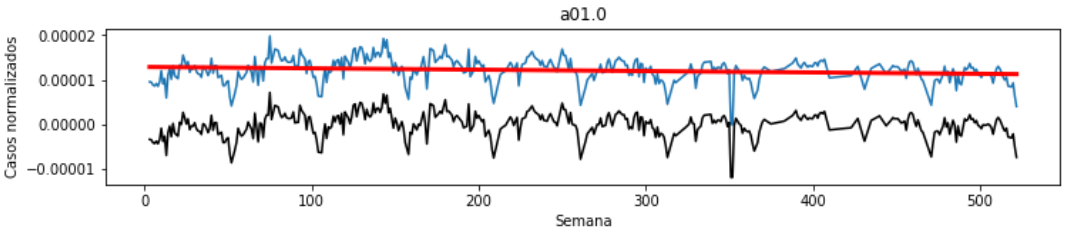

```
y = f(x) = -1.7653075450894884e-06 x + 0.0019754909758688734
error 1.215881678974069e-07
p =  1.0774404716157897e-37
pendiente significativa
R^2 0.3765563116502911
```

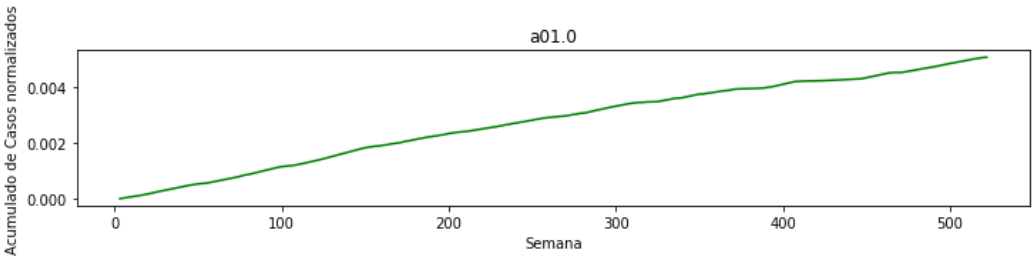


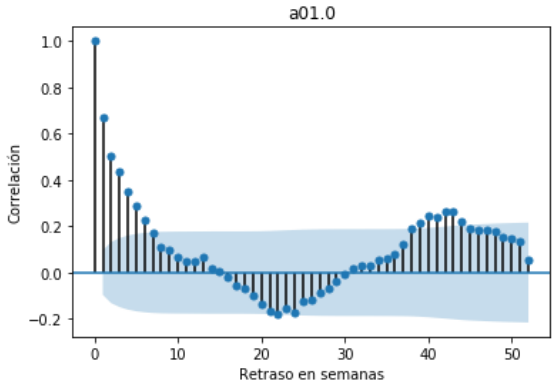




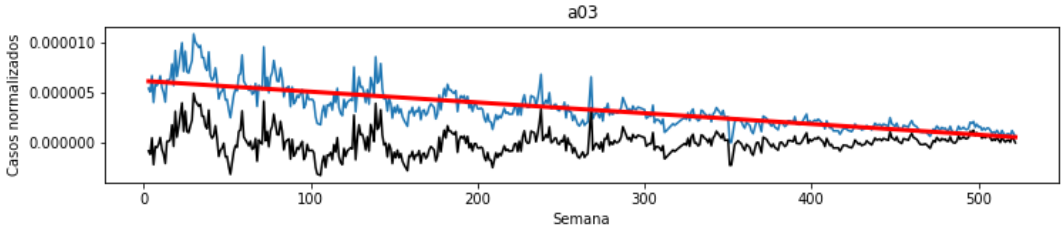

```
y = f(x) = -3.0752338979568535e-09 x + 1.2904965711605356e-05
error 8.36749265385657e-10
p =  0.0002687345236732541
pendiente significativa
R^2 0.03144815483703095
```

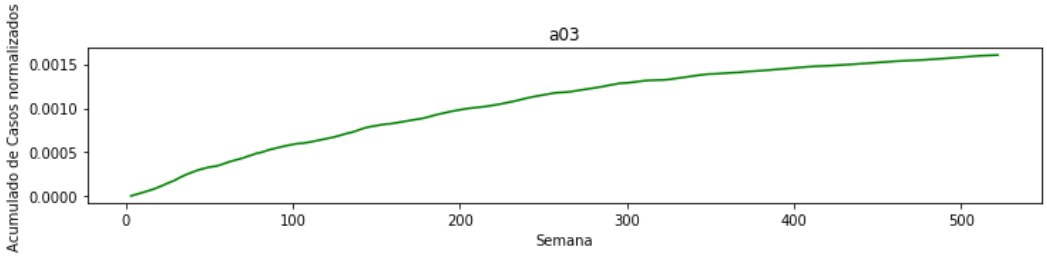


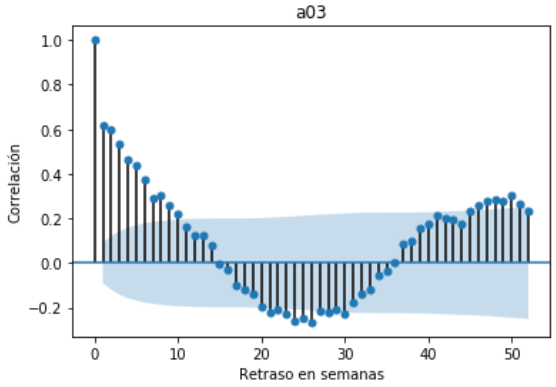





```
y = f(x) = -1.0785666679174943e-08 x + 6.155839207125693e-06
error 3.564598970127455e-10
p =  3.9269145731807006e-112
pendiente significativa
R^2 0.6622135065357393
```

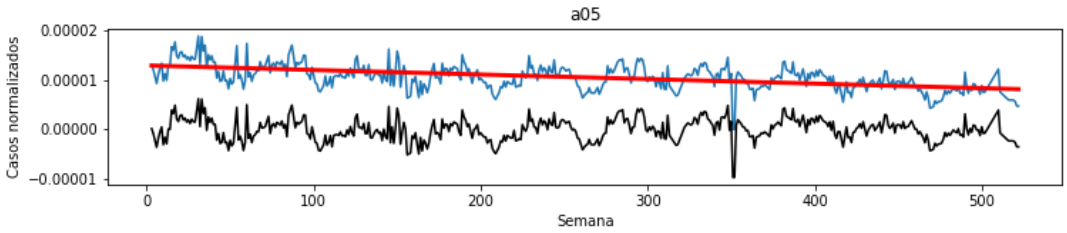


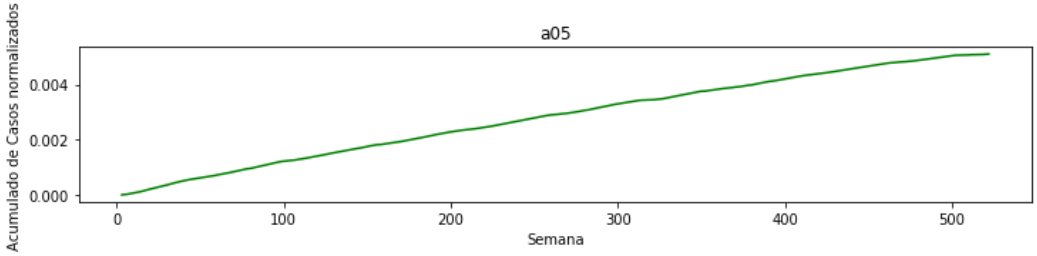


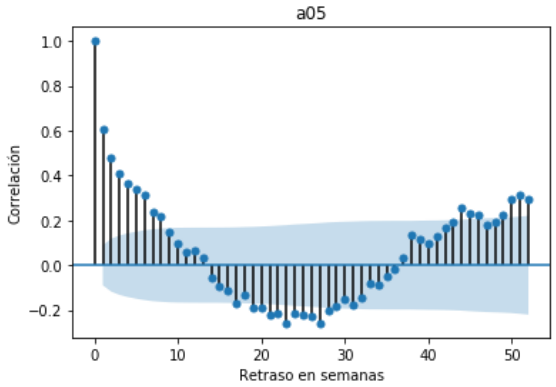



```
y = f(x) = -9.213574424729245e-09 x + 1.2957954379028175e-05
error 6.75300549213463e-10
p =  4.863591366039641e-36
```

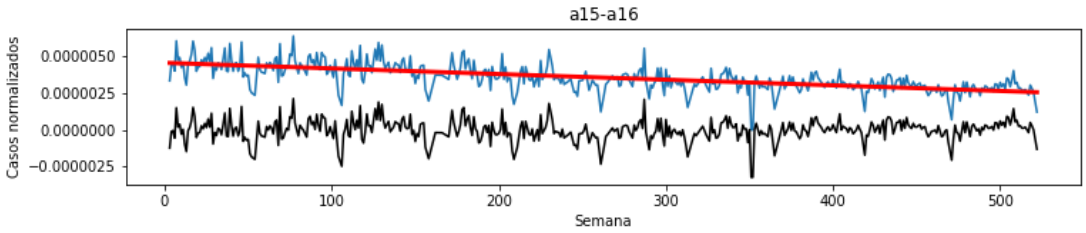
```
pendiente significativa
R^2 0.2798612445705242
```

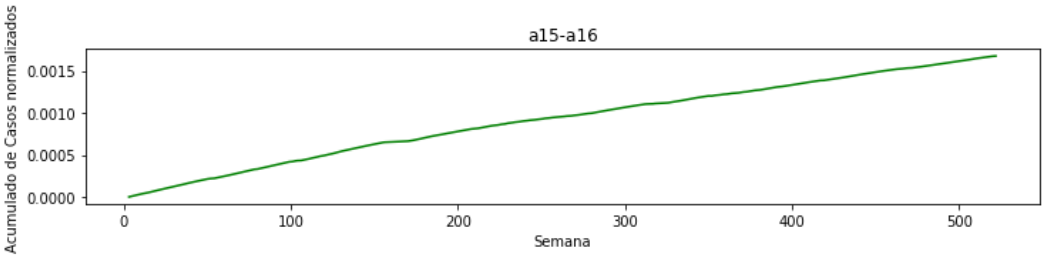


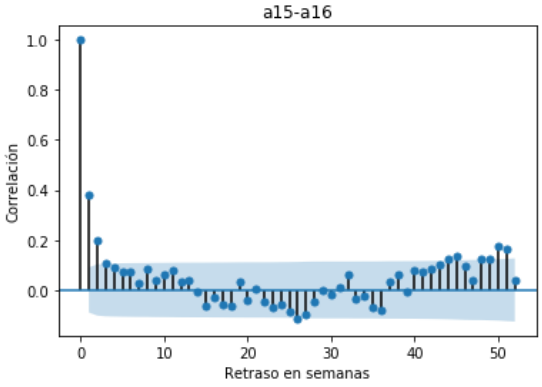




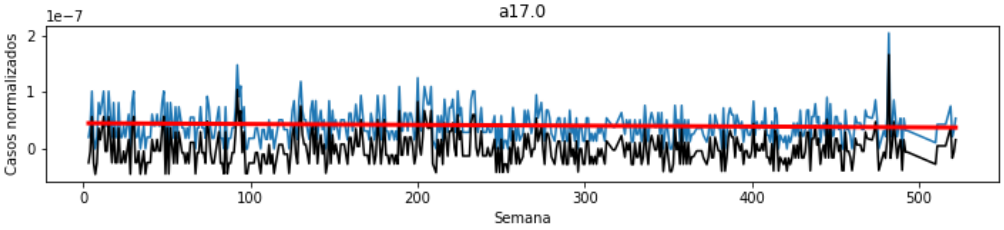

```
y = f(x) = -3.831012274635742e-09 x + 4.559110665311131e-06
error 2.056660576324692e-10
p =  2.1083769747172313e-58
pendiente significativa
R^2 0.4247090552255535
```

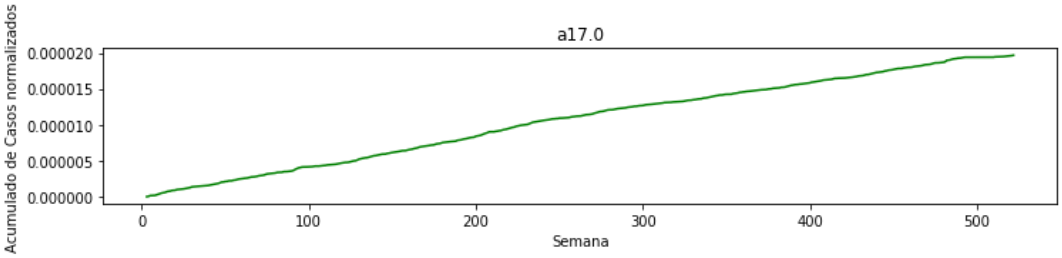


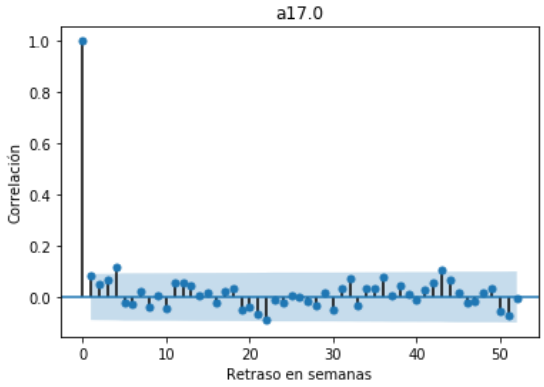




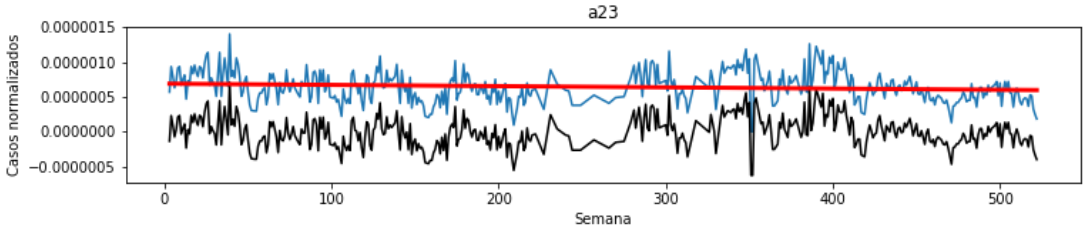

```
y = f(x) = -1.4627725963638185e-11 x + 4.5153183905681134e-08
error 8.6165080095987e-12
p =  0.09023493838847157
pendiente no significativa
R^2 0.006068835261027791
```

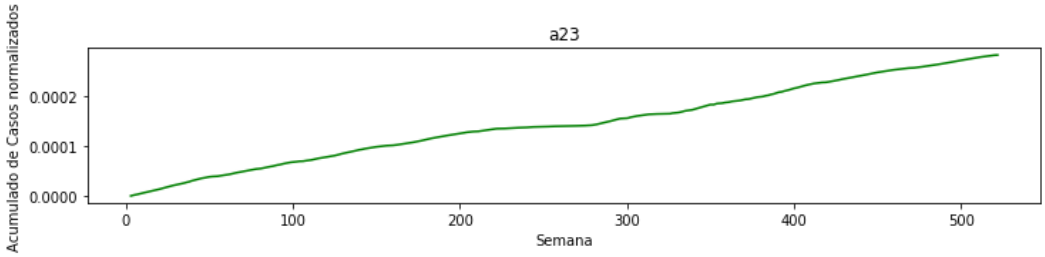


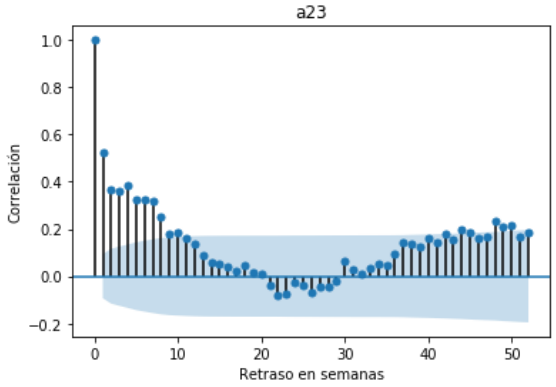





```
y = f(x) = -1.7642681104091792e-10 x + 6.894953499860091e-07
error 6.403728082254567e-11
p =  0.0061121040534305735
pendiente significativa
R^2 0.01699628644380042
```

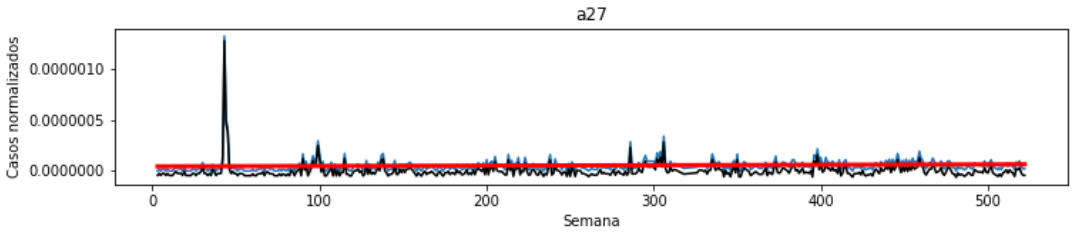


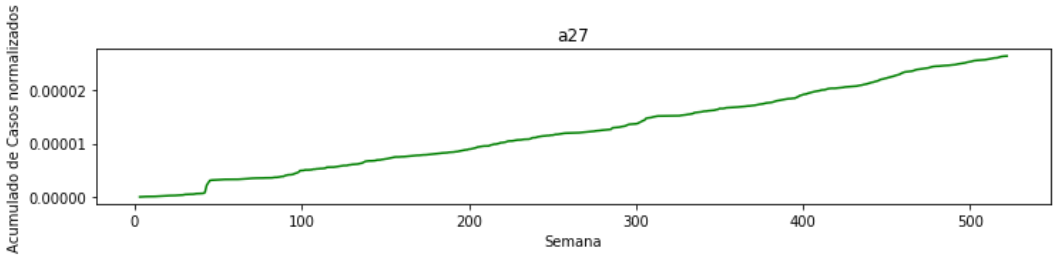


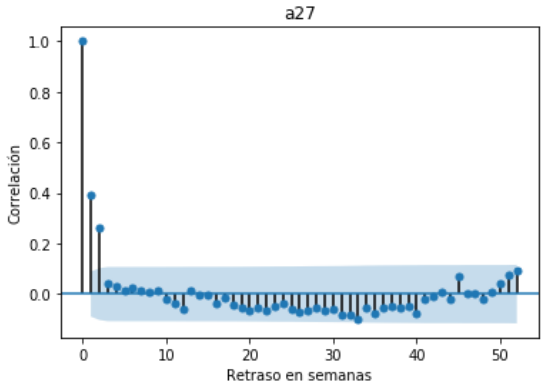



```
y = f(x) = 3.8938781393720075e-11 x + 4.4324399670136904e-08
error 2.301537019363991e-11
p =  0.09131901816285937
```

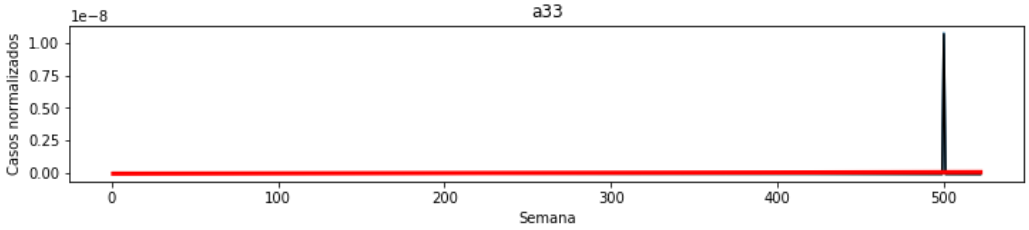
```
pendiente no significativa
R^2 0.00590350974760678
```

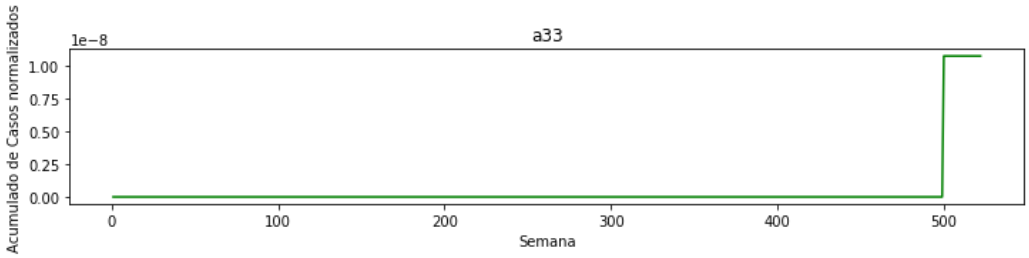


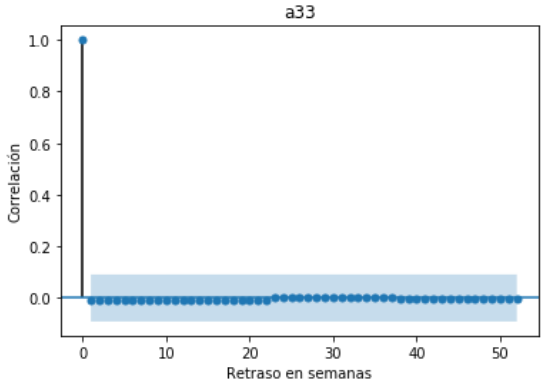




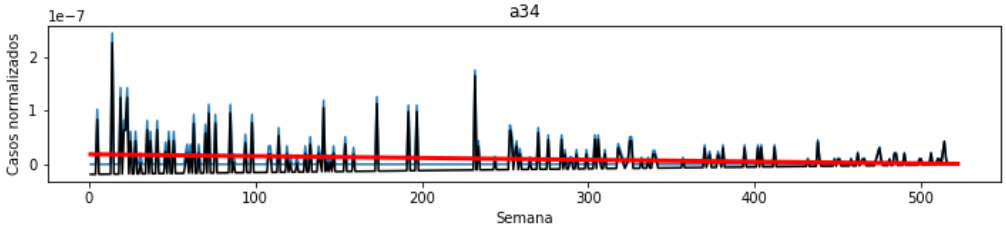

```
y = f(x) = 2.3434709729609935e-13 x + -3.6003147872535345e-11
error 1.478629682740248e-13
p =  0.1136746875979033
pendiente no significativa
R^2 0.005407586602284264
```

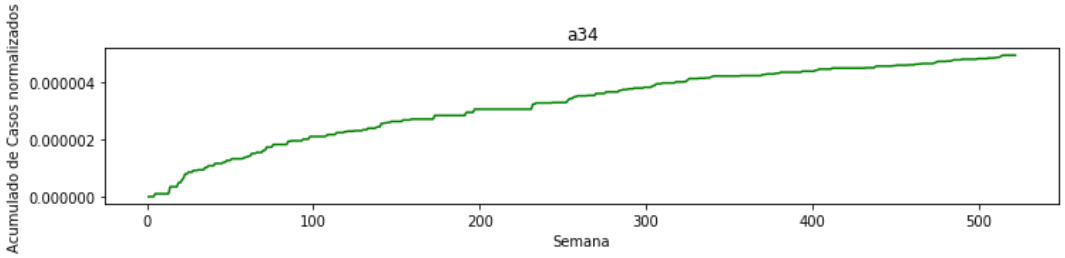


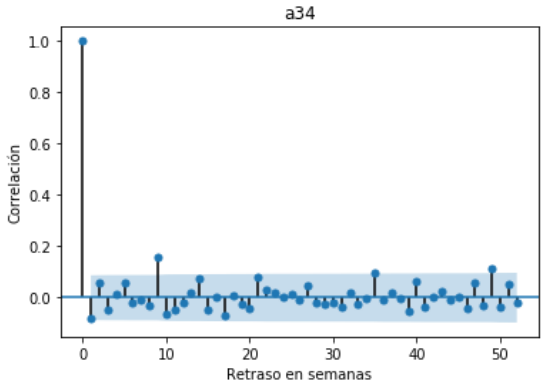




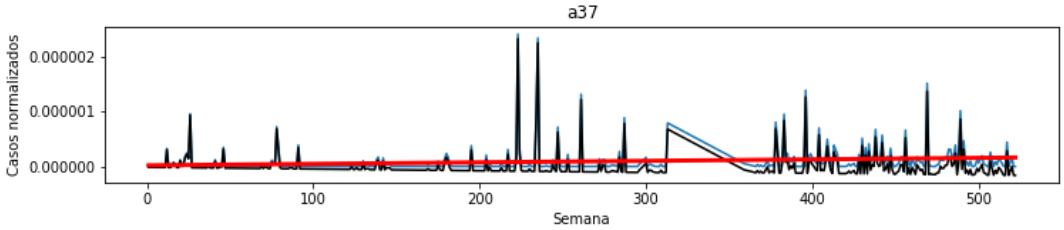

```
y = f(x) = -3.4234098194666664e-11 x + 1.8622107713377432e-08
error 7.415007265413988e-12
p =  4.945787391949576e-06
pendiente significativa
R^2 0.040422588588555235
```

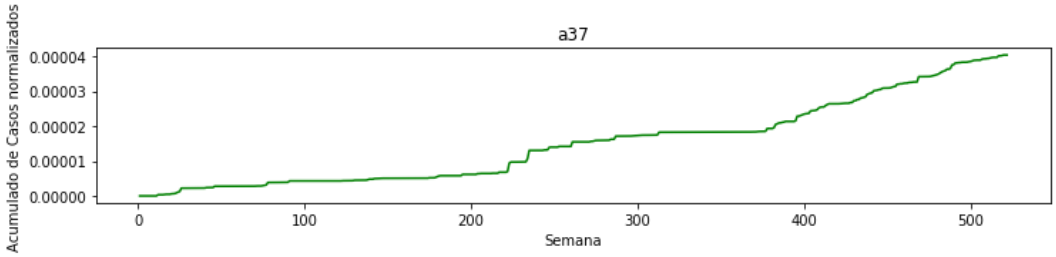


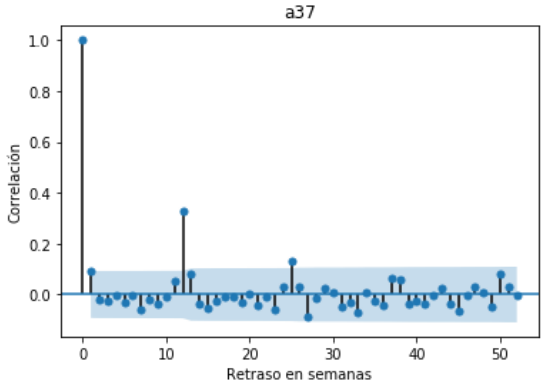





```
y = f(x) = 2.725400812299939e-10 x + 1.9471946148640167e-08
error 7.110945660238779e-11
p =  0.00014458146666075644
pendiente significativa
R^2 0.031274848966871695
```

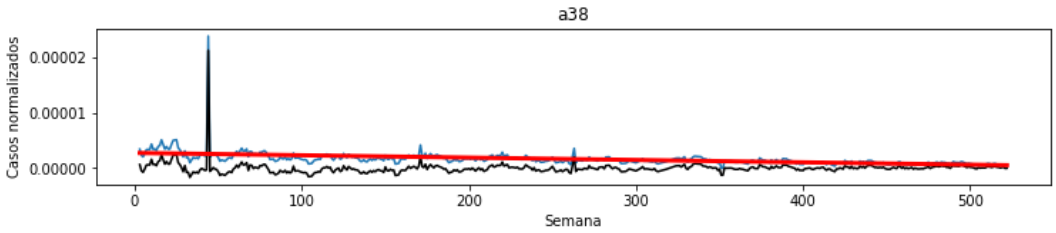


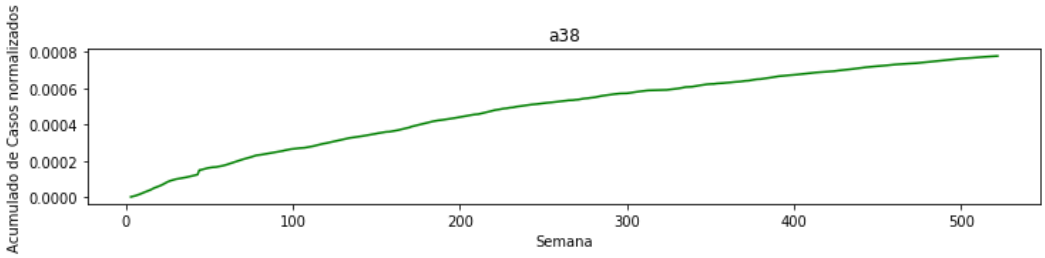


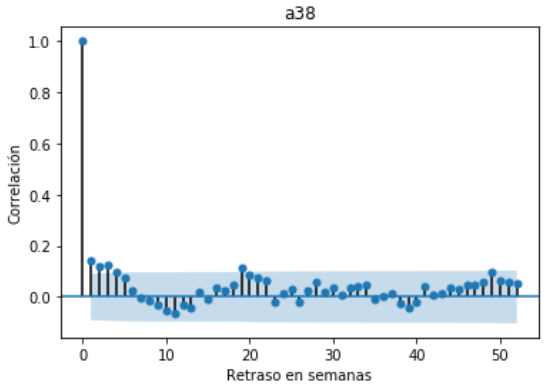



```
y = f(x) = -4.255464112013022e-09 x + 2.7470903702510135e-06
error 3.3548559373967543e-10
p =  6.366969283108536e-32
```

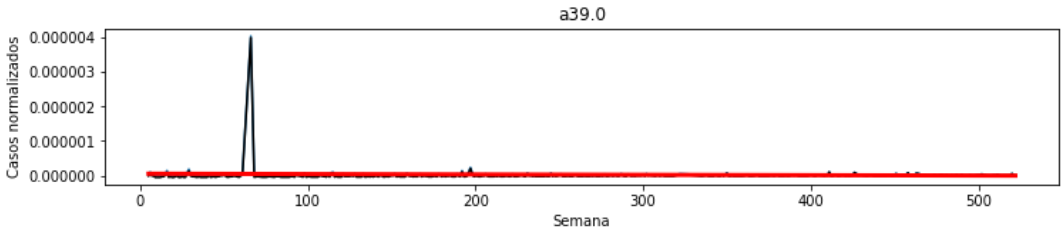
```
pendiente significativa
R^2 0.25462455709847065
```

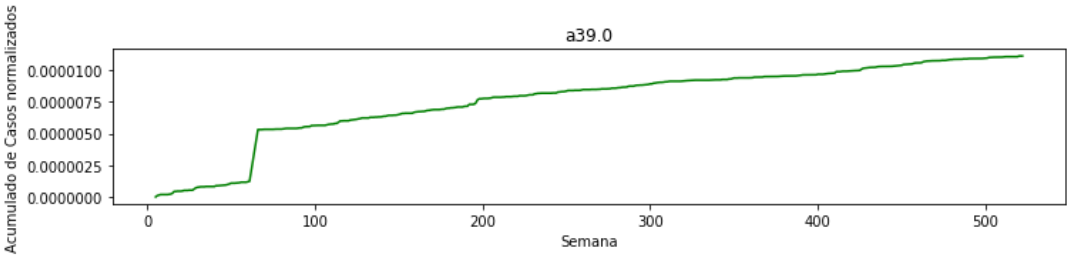


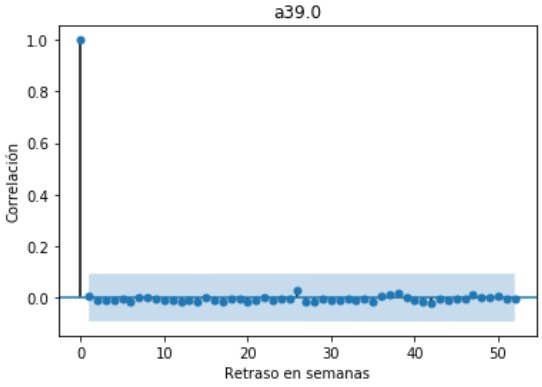




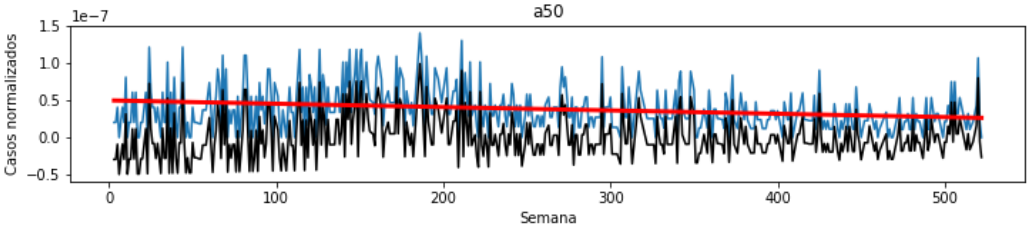

```
y = f(x) = -1.0744562226275755e-10 x + 5.3269706349552e-08
error 6.010845348981309e-11
p =  0.07452781641973526
pendiente no significativa
R^2 0.007081769917450651
```

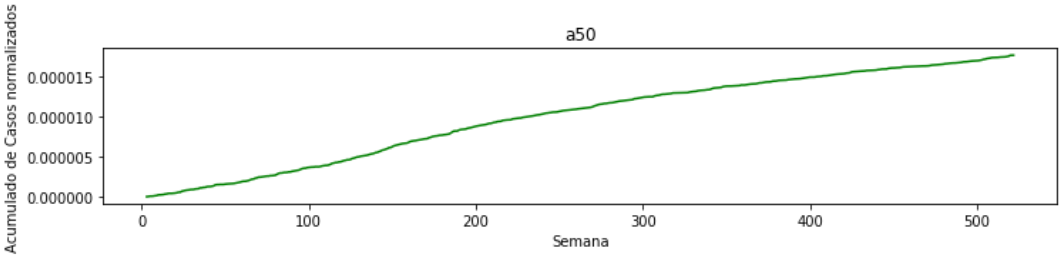


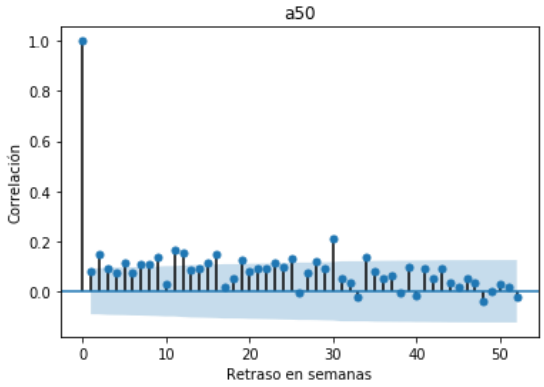




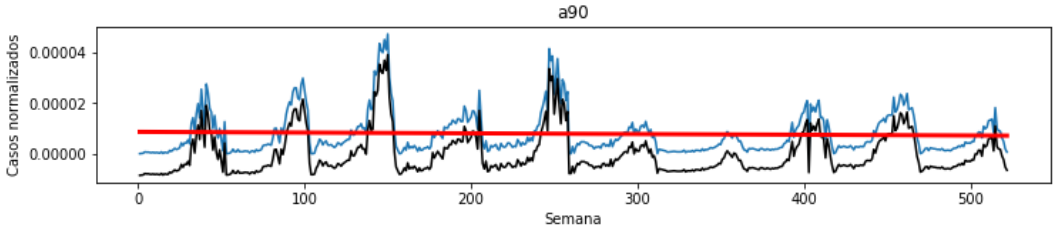

```
y = f(x) = -4.51442956629053e-11 x + 5.003109663849265e-08
error 8.825884482273087e-12
p =  4.625709147491021e-07
pendiente significativa
R^2 0.05414963353204715
```

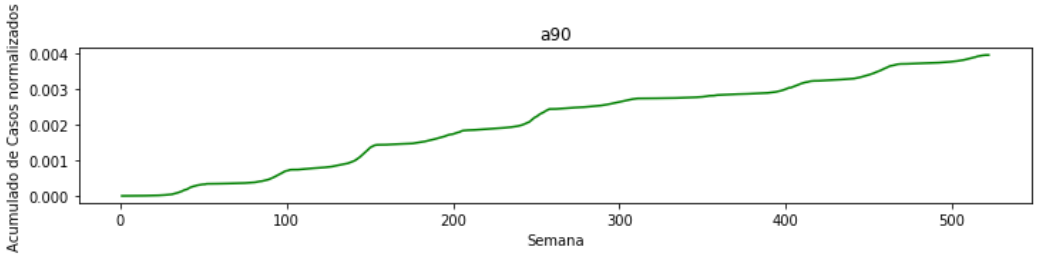


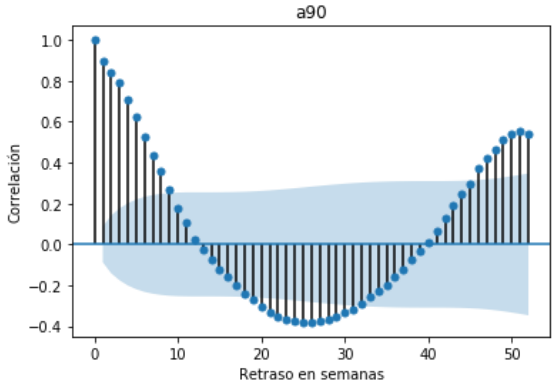





```
y = f(x) = -2.8757621426513095e-09 x + 8.65805695809489e-06
error 2.5195362618720955e-09
p =  0.25425697912167106
pendiente no significativa
R^2 0.002603944949826924
```

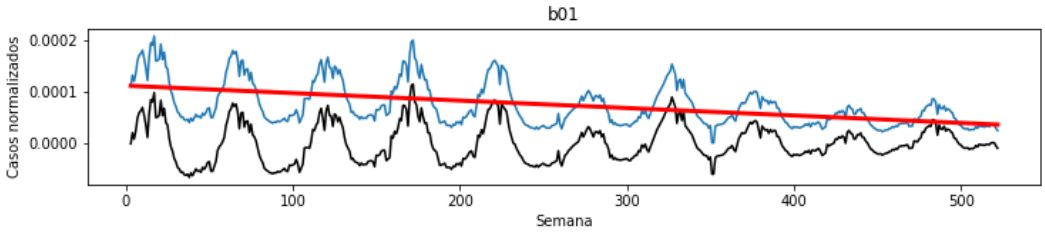


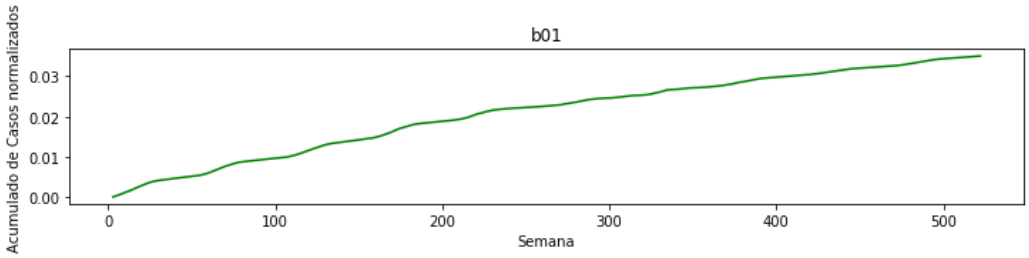


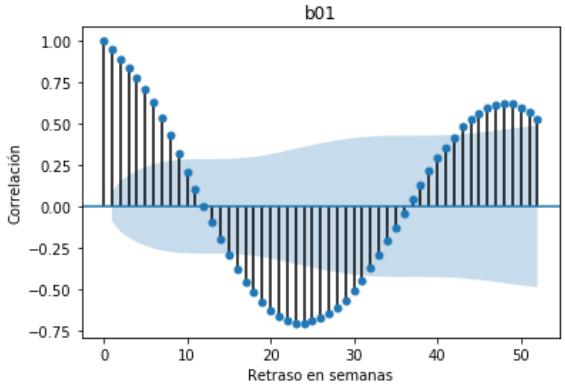



```
y = f(x) = -1.4644681914017878e-07 x + 0.00011211051217541277
error 1.1228826202874234e-08
p =  2.0750705342062423e-33
```

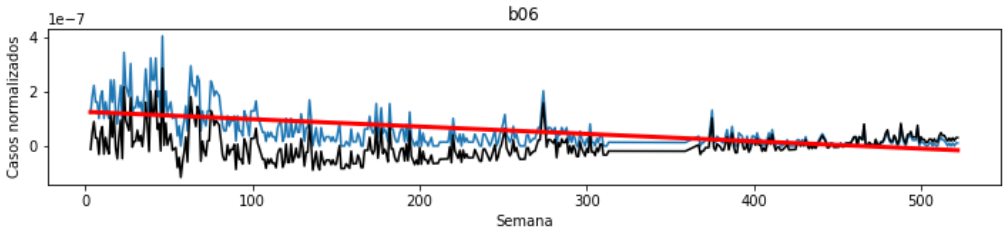
```
pendiente significativa
R^2 0.2653193213169377
```

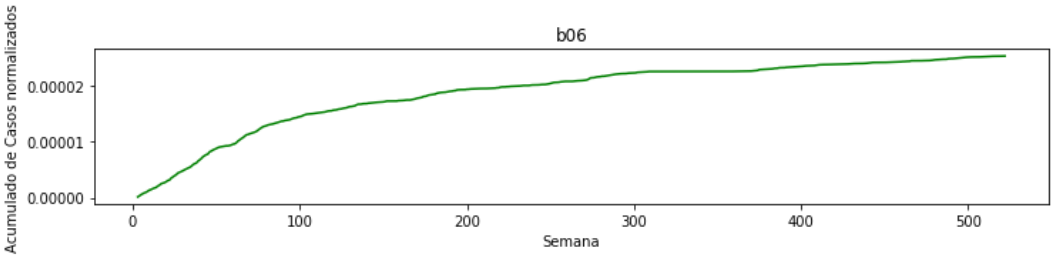


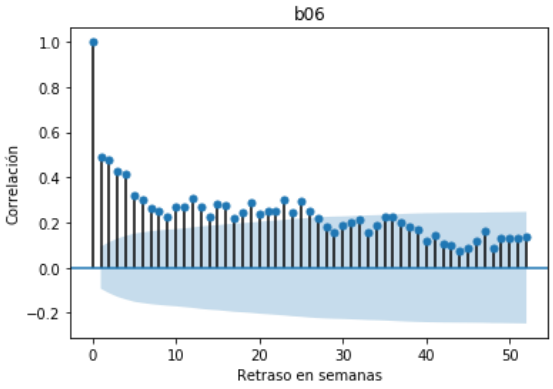




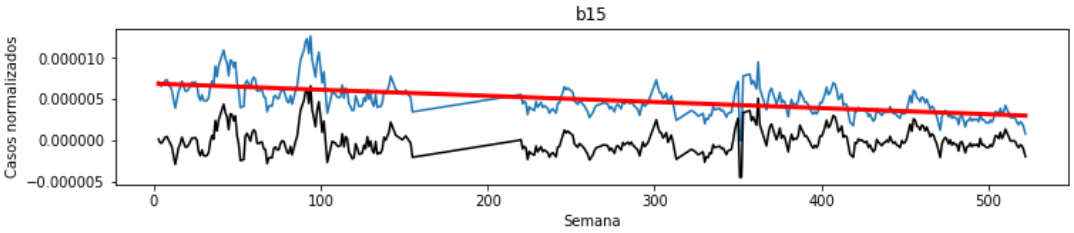

```
y = f(x) = -2.722125710622442e-10 x + 1.262294141156984e-07
error 1.588658069624221e-11
p =  7.30165010543899e-51
pendiente significativa
R^2 0.3985877970692943
```

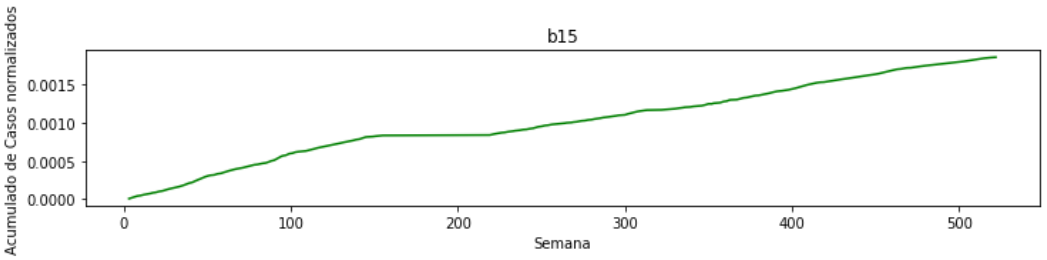


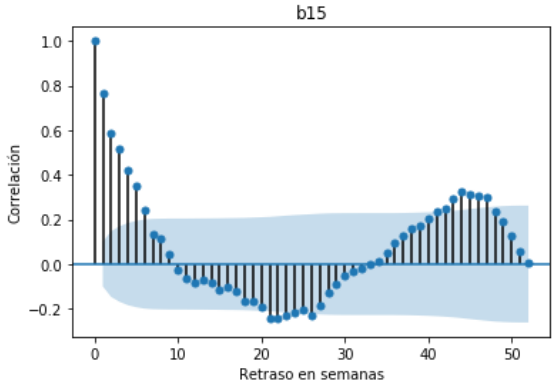




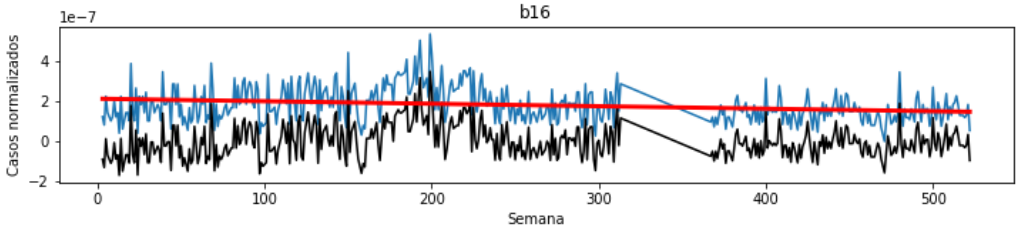

```
y = f(x) = -7.433434732564669e-09 x + 6.901511580636186e-06
error 4.809110869935852e-10
p =  3.1523565347787724e-42
pendiente significativa
R^2 0.3841635208468614
```

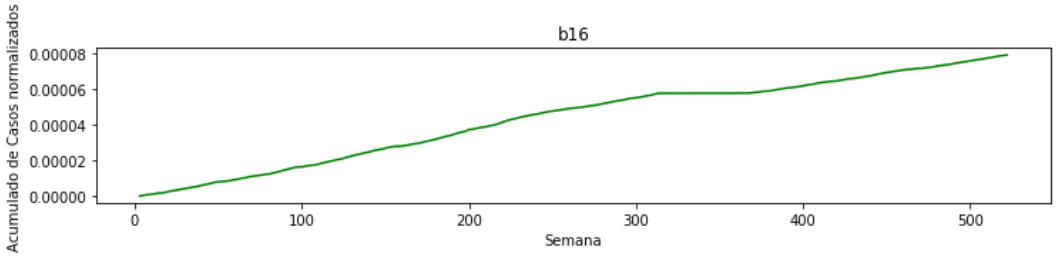


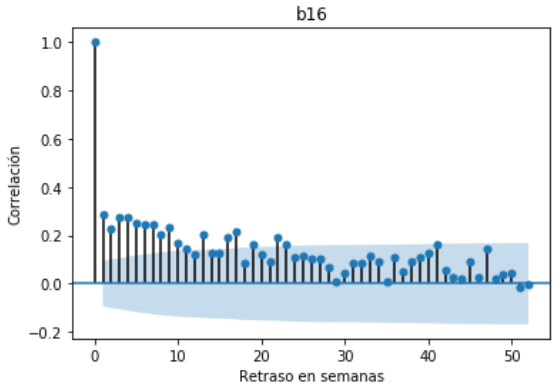





```
y = f(x) = -1.2411343091809244e-10 x + 2.1160165305354125e-07
error 2.468137006599137e-11
p =  7.217507598355145e-07
pendiente significativa
R^2 0.05458192227303572
```

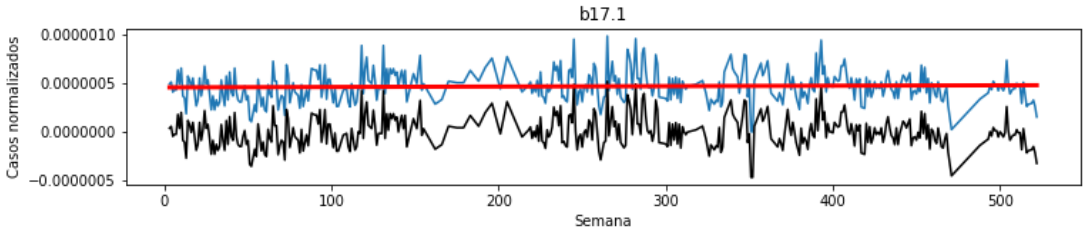


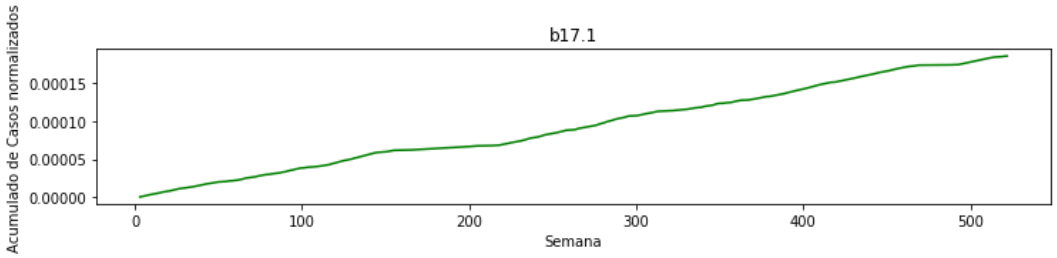


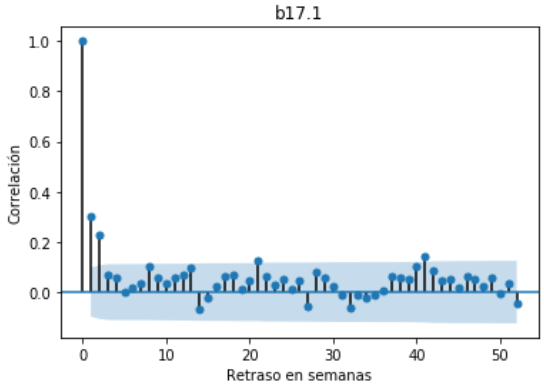



```
y = f(x) = 4.306189172696877e-11 x + 4.5311776302657916e-07
error 4.983732447188263e-11
p =  0.3880813876059074
```

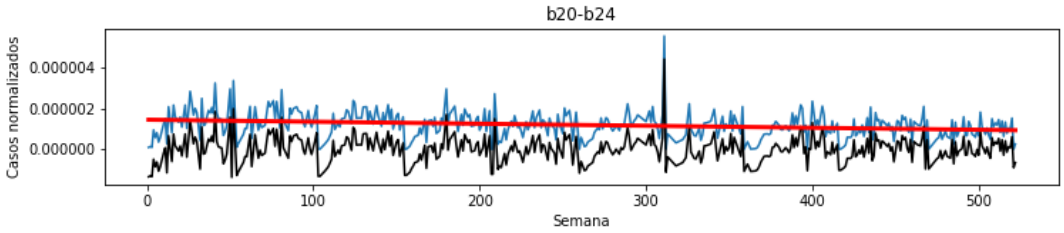
```
pendiente no significativa
R^2 0.0018723188046217538
```

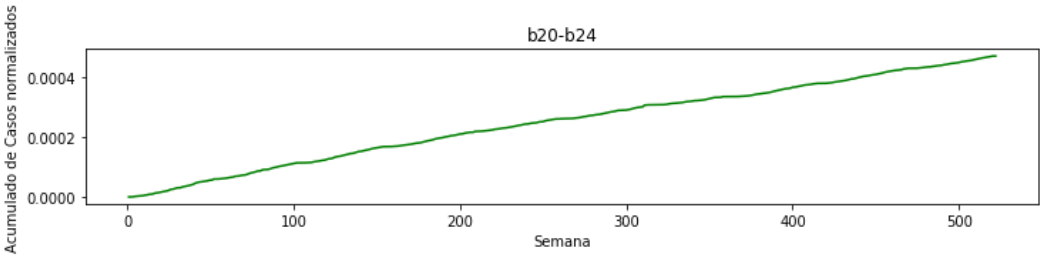


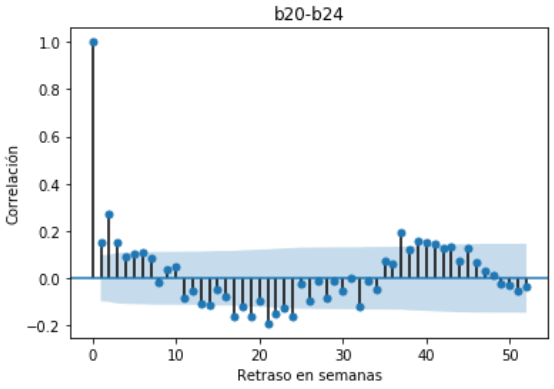




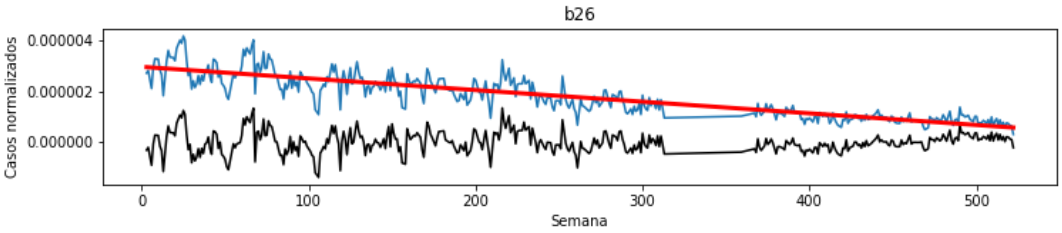

```
y = f(x) = -9.851452950958728e-10 x + 1.4372871687382767e-06
error 2.0422528814979408e-10
p =  2.004159852045715e-06
pendiente significativa
R^2 0.05484537751521497
```

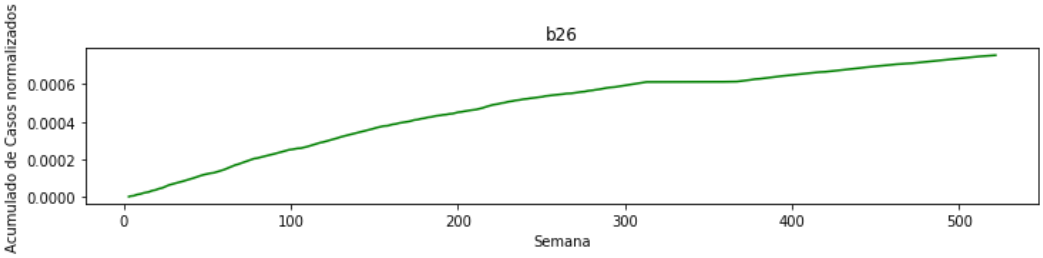


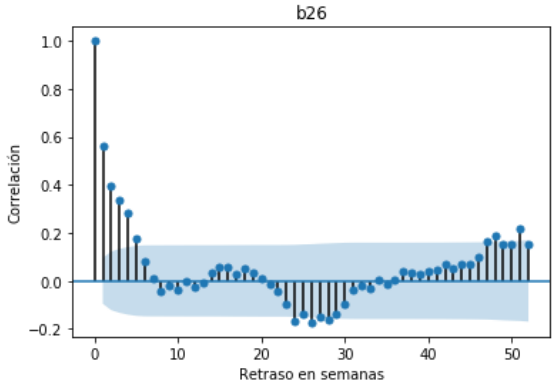




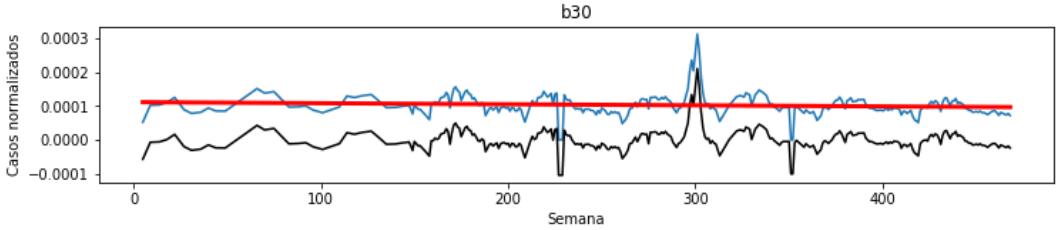

```
y = f(x) = -4.5977874526241305e-09 x + 2.970356116391645e-06
error 1.2200834000344058e-10
p =  5.731116914266081e-137
pendiente significativa
R^2 0.7713322815069262
```

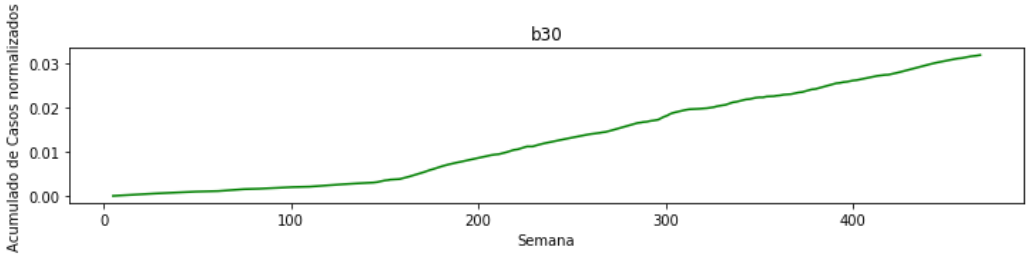


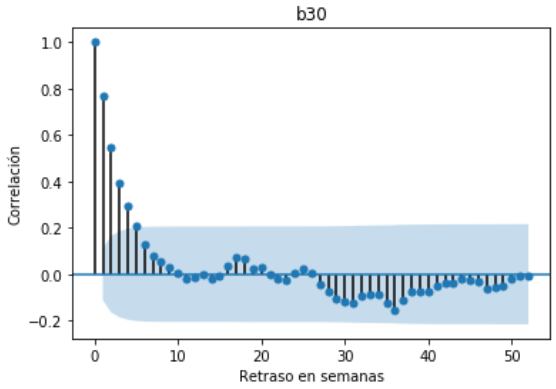





```
y = f(x) = -3.141197049928086e-08 x + 0.00011159382796415527
error 1.555599789385595e-08
p =  0.04432369209037648
pendiente significativa
R^2 0.013065683803978847
```

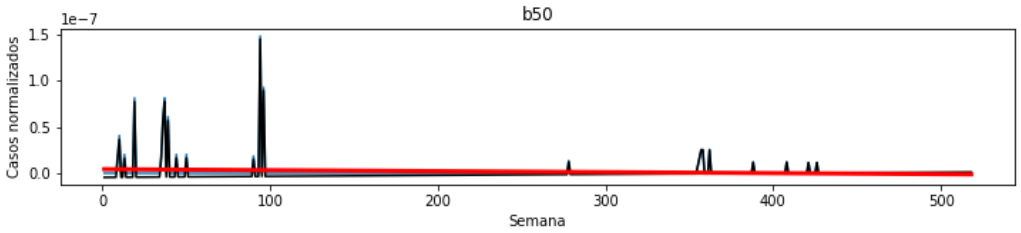


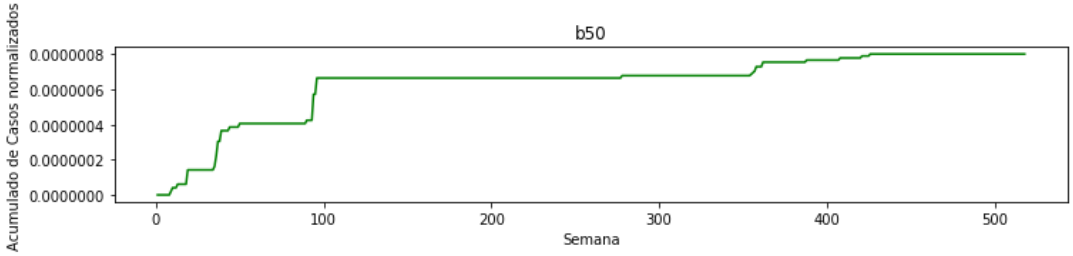


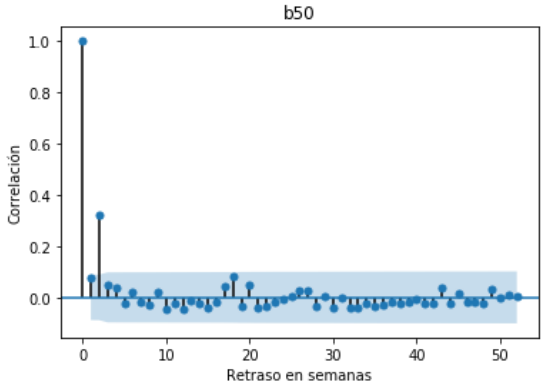



```
y = f(x) = -1.0673850709576887e-11 x + 4.355250762325799e-09
error 3.1880871555405942e-12
p =  0.0008764994132686745
```

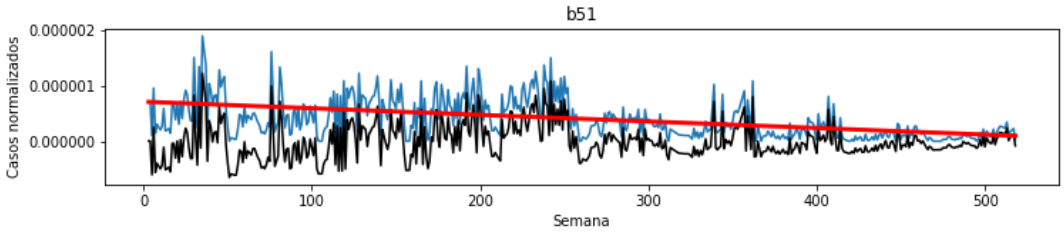
```
pendiente significativa
R^2 0.02236467960180641
```

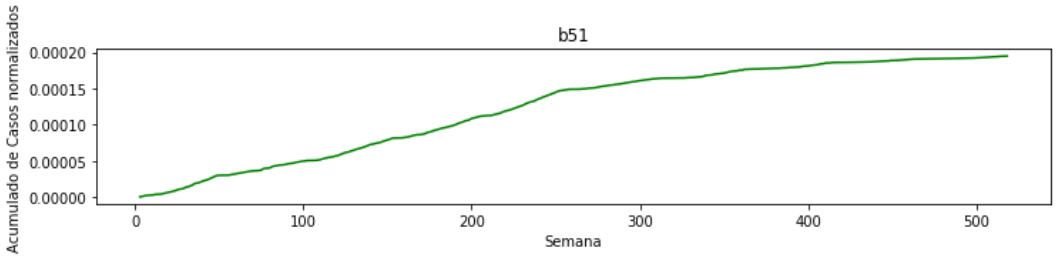


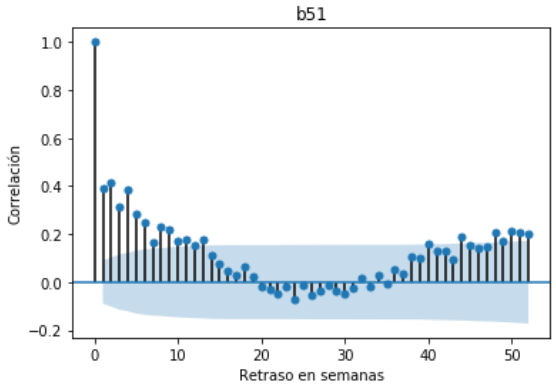




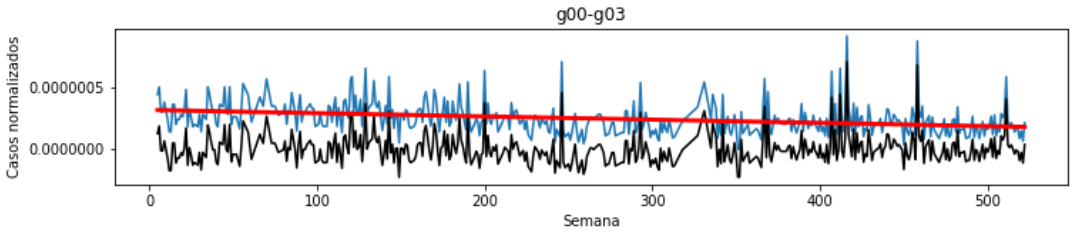

```
y = f(x) = -1.1834633525276088e-09 x + 7.100912129285122e-07
error 9.393463801467669e-11
p =  1.2559331815198554e-31
pendiente significativa
R^2 0.24928894104283042
```

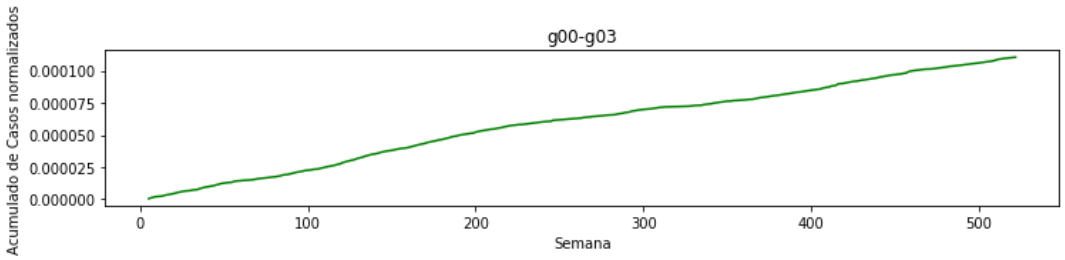


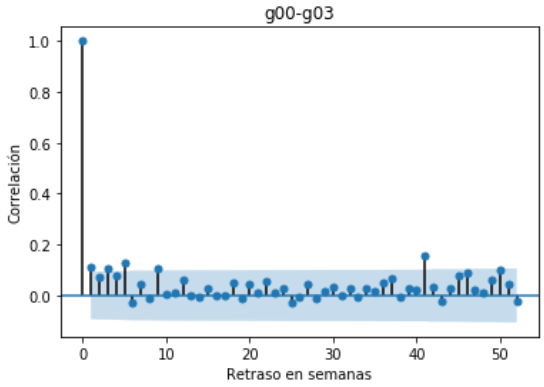




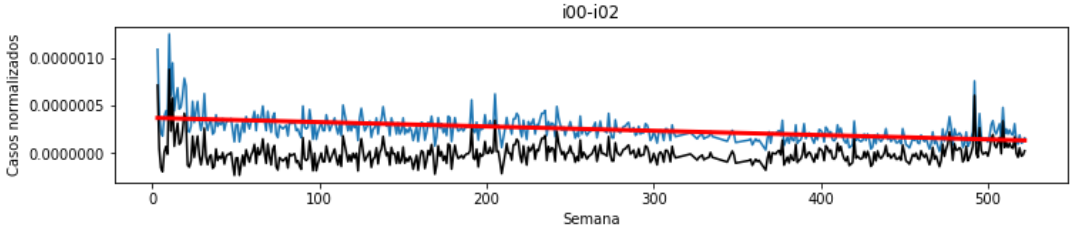

```
y = f(x) = -2.6544241562890327e-10 x + 3.2049521522763566e-07
error 3.7319118941308474e-11
p =  4.610321182741757e-12
pendiente significativa
R^2 0.10270484494415096
```

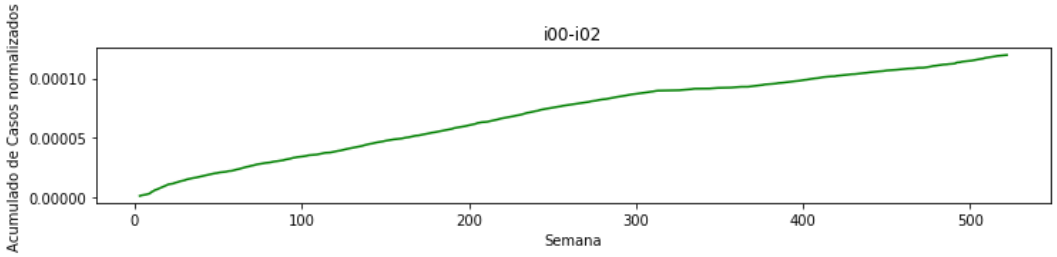


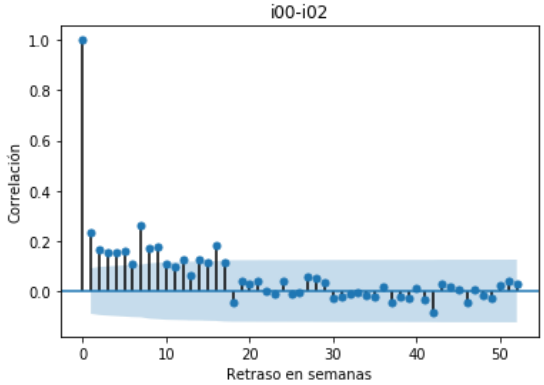





```
y = f(x) = -4.556172046800565e-10 x + 3.7500114147077754e-07
error 3.423510008372152e-11
p =  1.8548314561638924e-34
pendiente significativa
R^2 0.27626166419878756
```

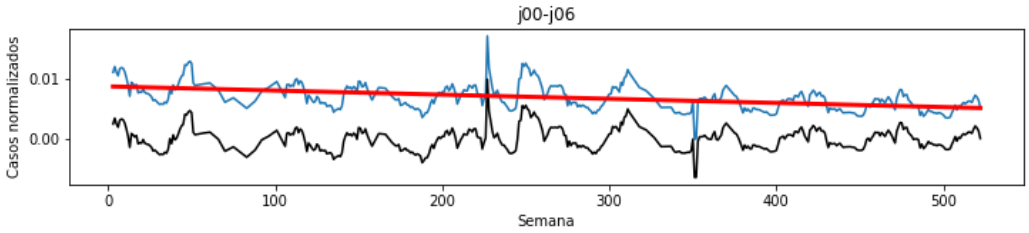


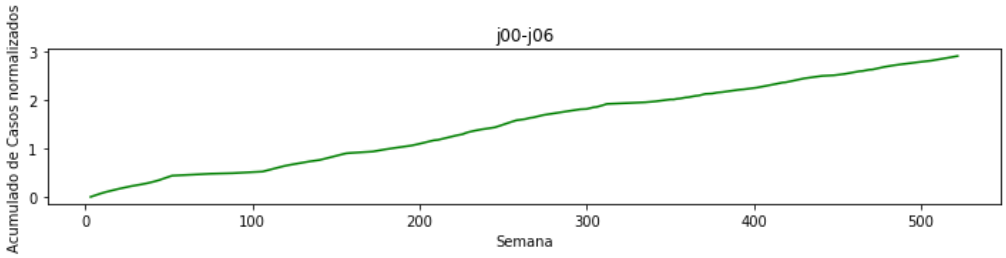


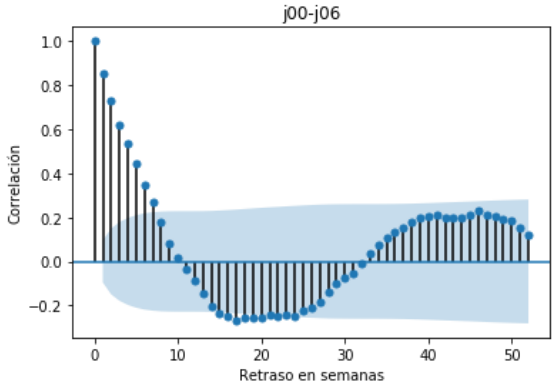



```
y = f(x) = -6.932314608726674e-06 x + 0.008805967823835786
error 6.162044690299544e-07
p =  7.503334430143699e-26
```

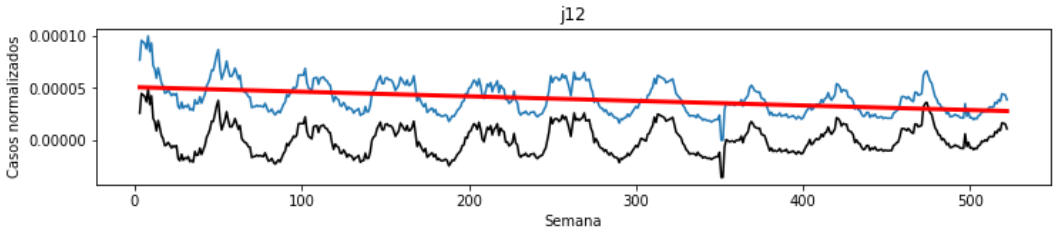
```
pendiente significativa
R^2 0.23113873276962404
```

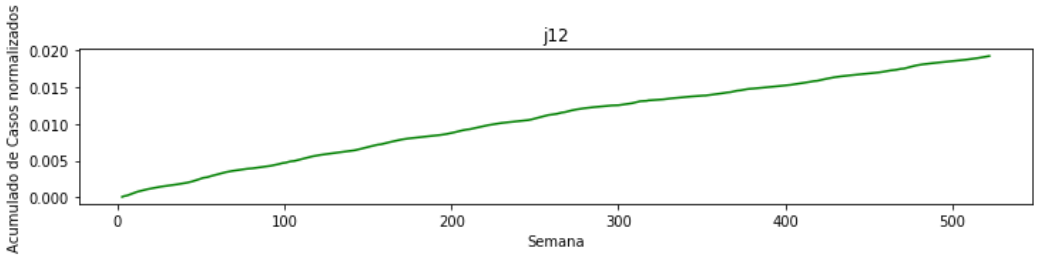


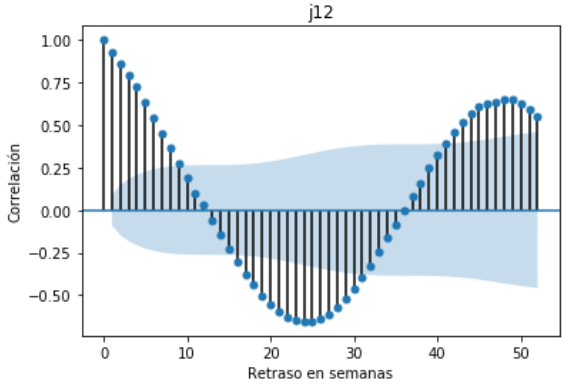




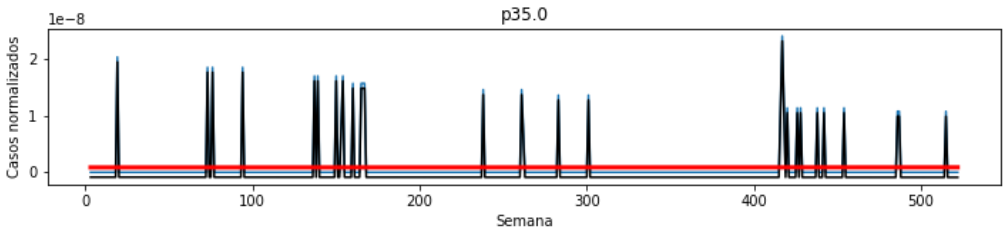

```
y = f(x) = -4.3876585629436324e-08 x + 5.08928603405923e-05
error 4.181979380800392e-09
p =  2.3727736779313174e-23
pendiente significativa
R^2 0.1846709286259034
```

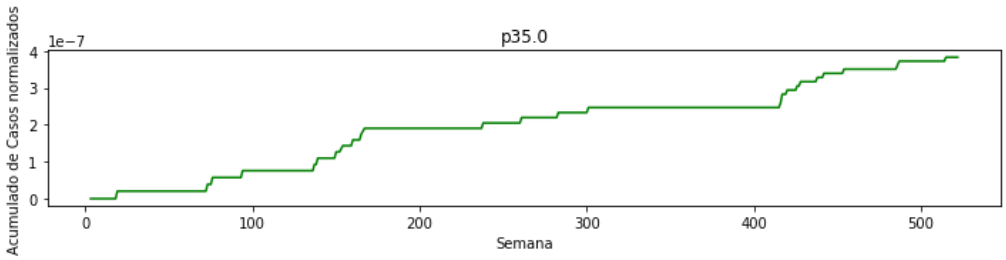


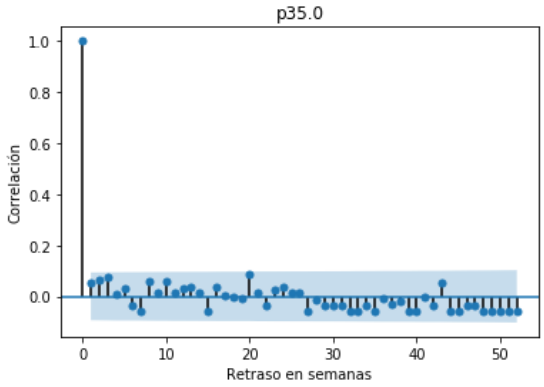




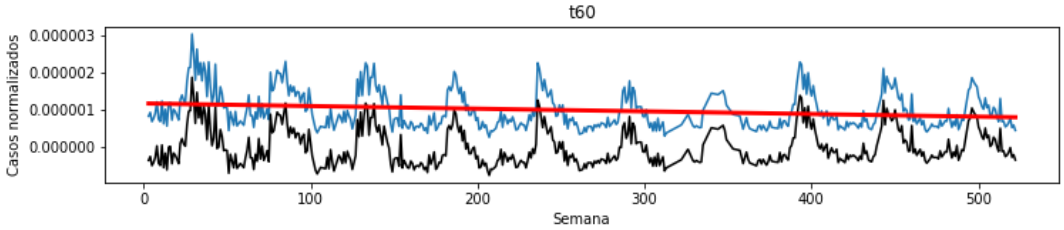

```
y = f(x) = 3.0607452607643954e-14 x + 8.632235917582137e-10
error 1.081125720208934e-12
p =  0.977427221531179
pendiente no significativa
R^2 1.8298987789071008e-06
```

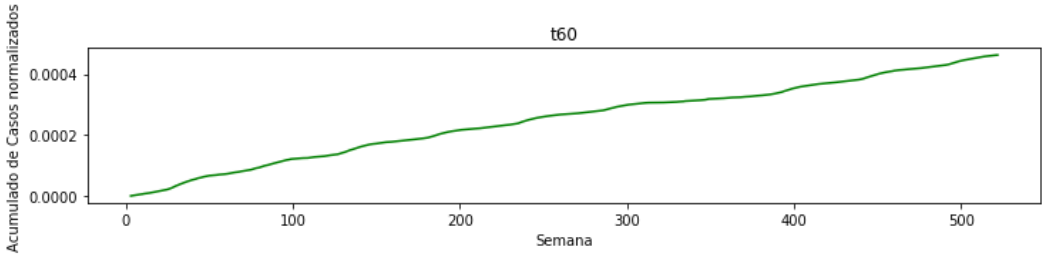


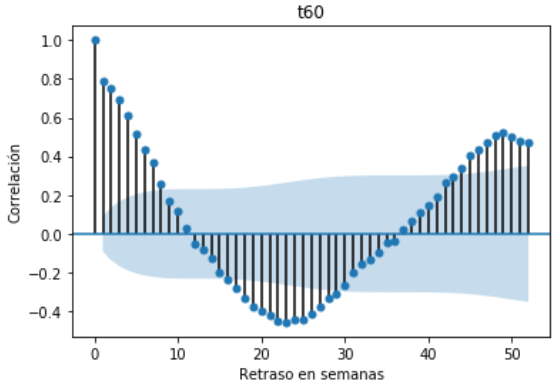





```
y = f(x) = -7.142618310354189e-10 x + 1.178302593061399e-06
error 1.457944295445013e-10
p =  1.332329783777257e-06
pendiente significativa
R^2 0.04918267323938655
```

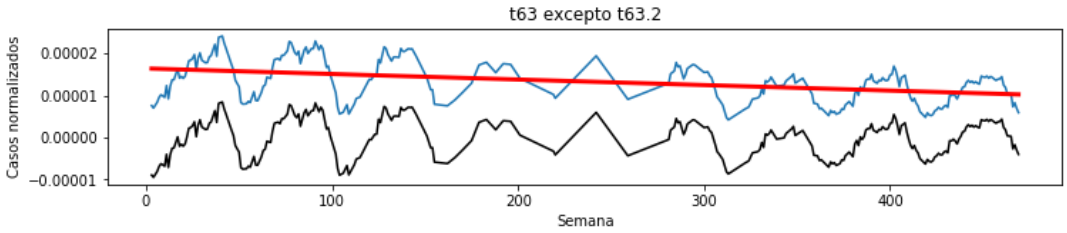


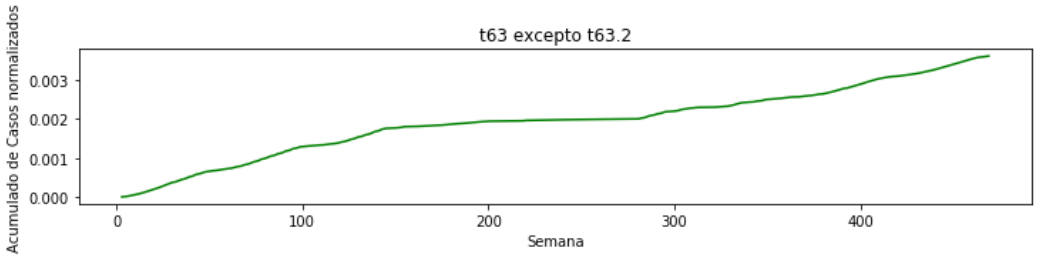


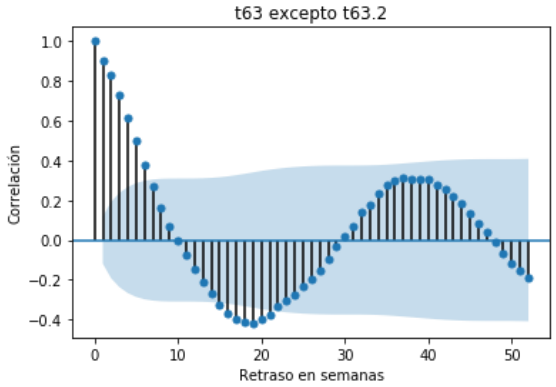



```
y = f(x) = -1.3158987800860395e-08 x + 1.643445025799908e-05
error 1.6084785128648306e-09
p =  1.0885896084621633e-14
```

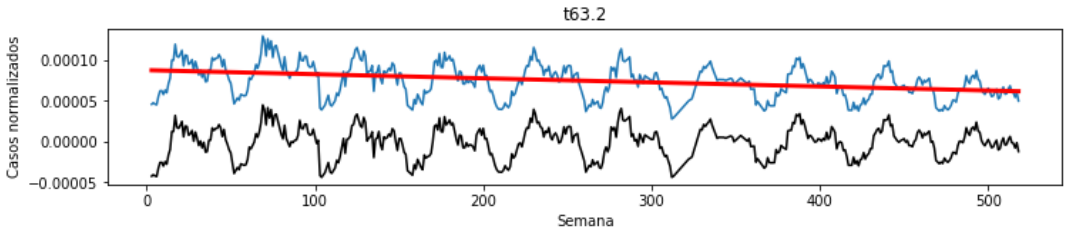
```
pendiente significativa
R^2 0.19747207526346663
```

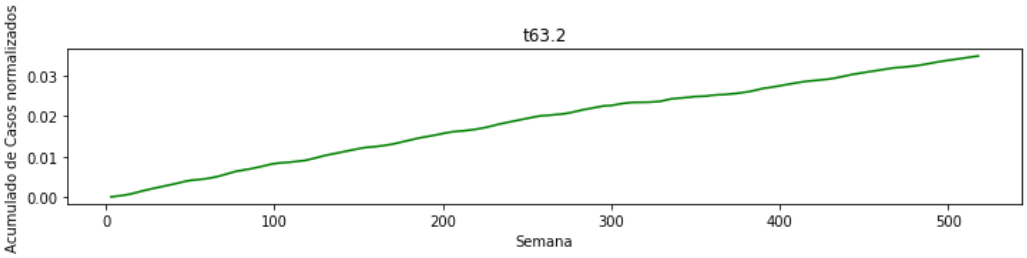


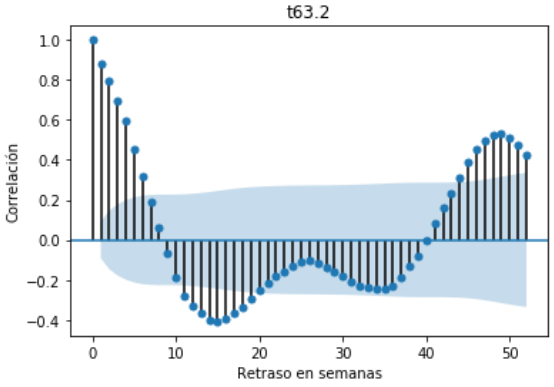




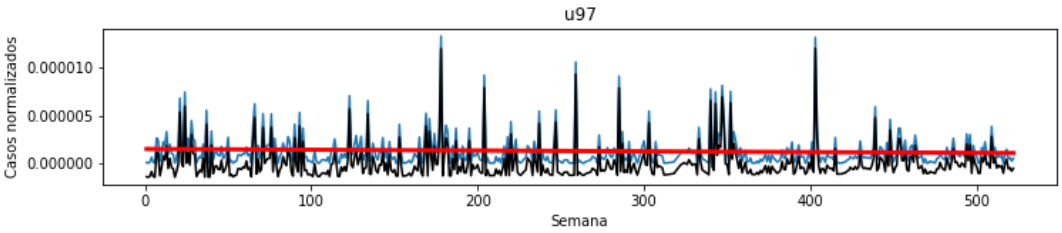

```
y = f(x) = -5.001283641779608e-08 x + 8.79623291848092e-05
error 5.714736801137942e-09
p =  3.9731738436330694e-17
pendiente significativa
R^2 0.14220420601744926
```

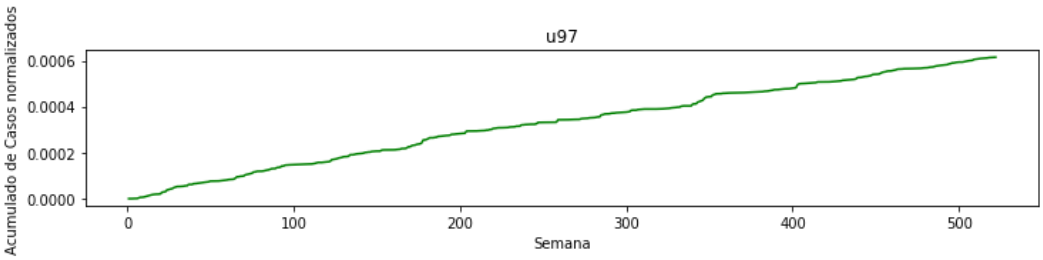


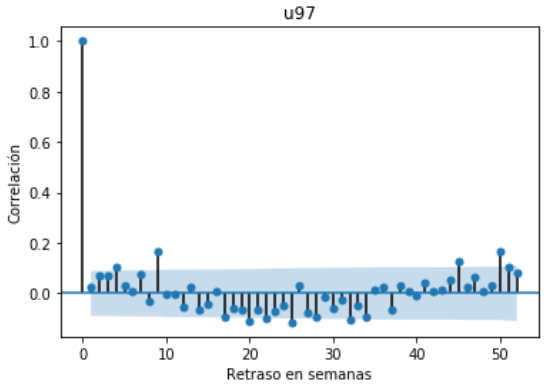




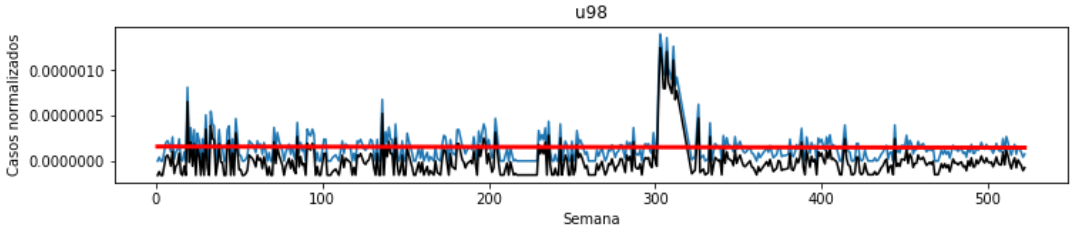

```
y = f(x) = -8.130421944150189e-10 x + 1.4827816461880552e-06
error 5.134961166576471e-10
p =  0.11399979312553642
pendiente no significativa
R^2 0.005185003922499099
```

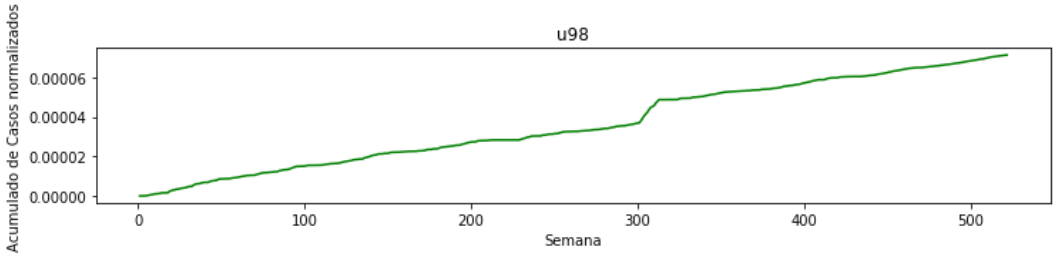


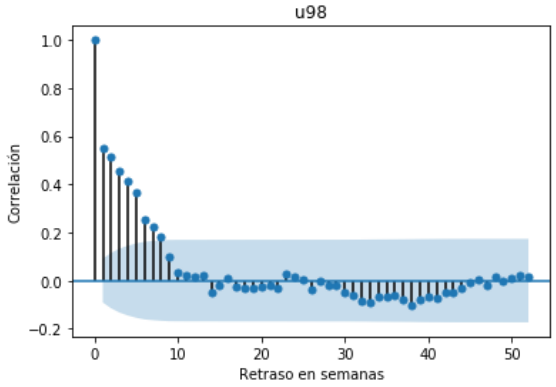





```
y = f(x) = -2.3438815029578847e-11 x + 1.5962437413039917e-07
error 5.628337741371575e-11
p =  0.6772793134611802
pendiente no significativa
R^2 0.00037523732655554817
```

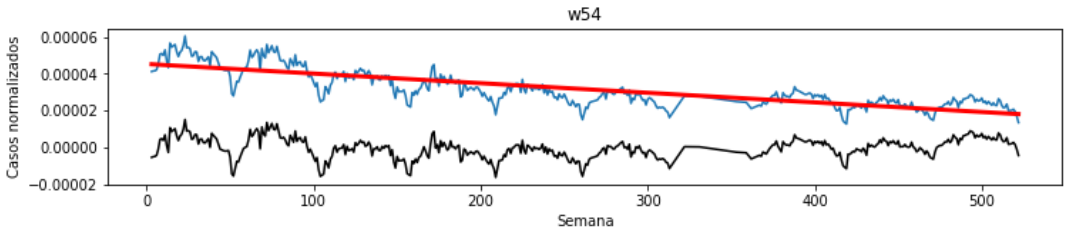


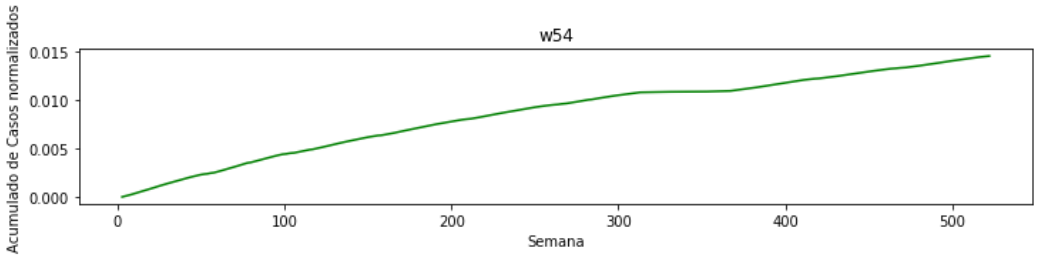


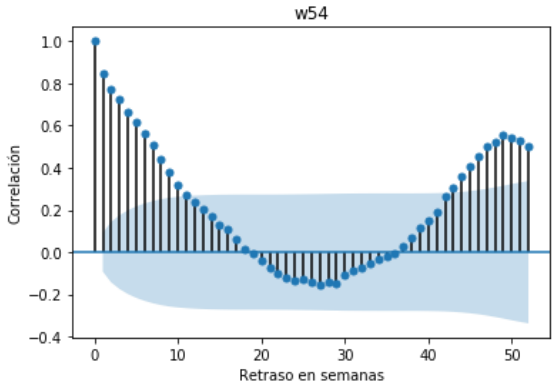



```
y = f(x) = -5.193298199318447e-08 x + 4.533358442590547e-05
error 1.6692637054937939e-09
p =  1.6078008523125964e-114
```

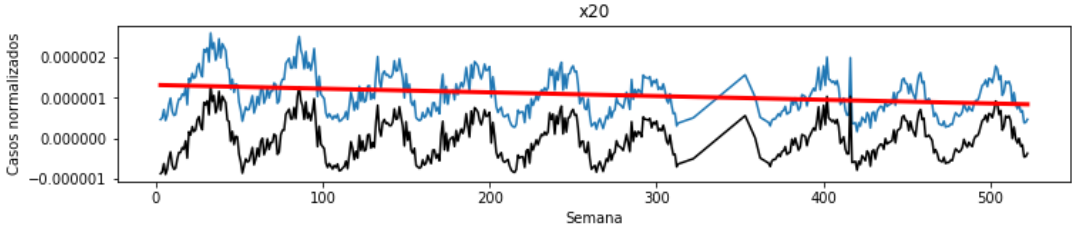
```
pendiente significativa
R^2 0.6811910446622308
```

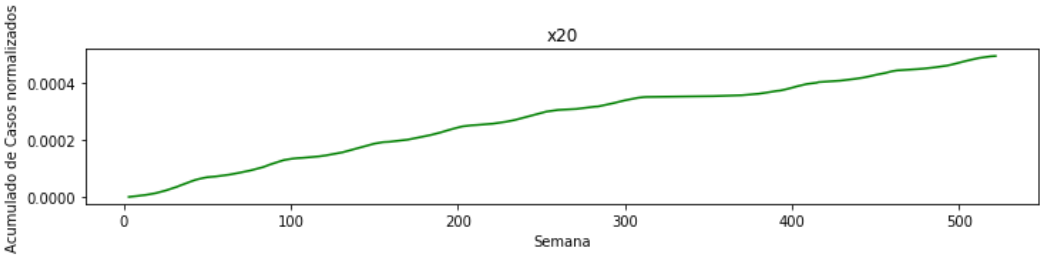


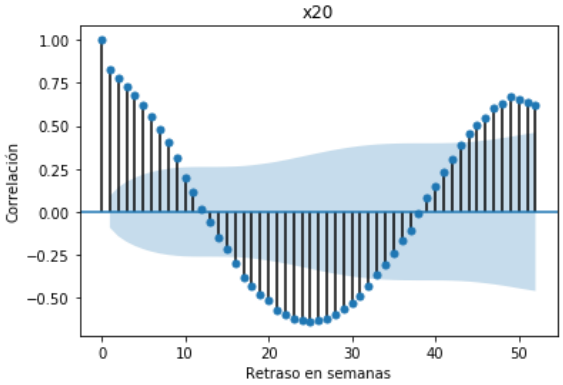




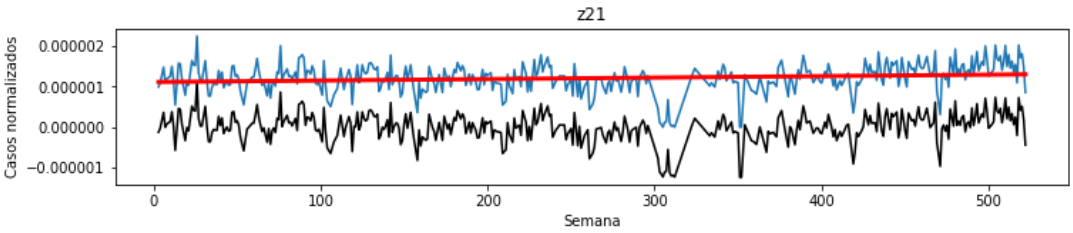

```
y = f(x) = -9.132331371853938e-10 x + 1.3202325787207294e-06
error 1.431149104840528e-10
p =  4.359158383428425e-10
pendiente significativa
R^2 0.08264080526994147
```

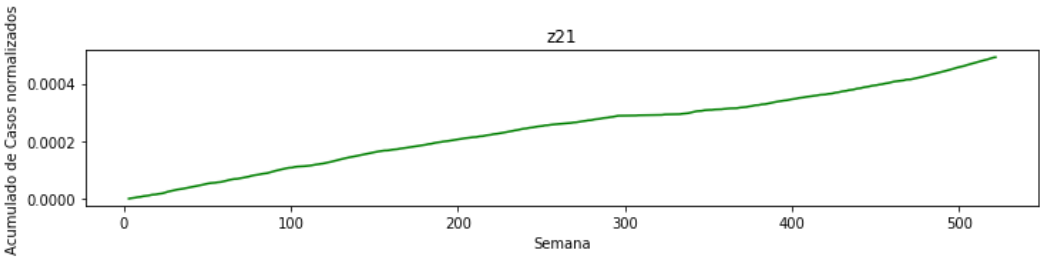


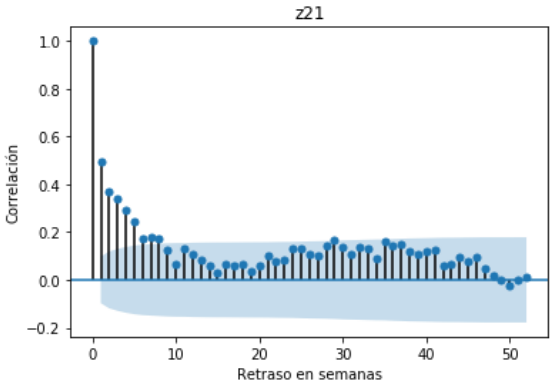





```
y = f(x) = 3.638834463785146e-10 x + 1.1174278631513553e-06
error 1.1201987356227443e-10
p =  0.001257639893824037
pendiente significativa
R^2 0.025453964627936122
```

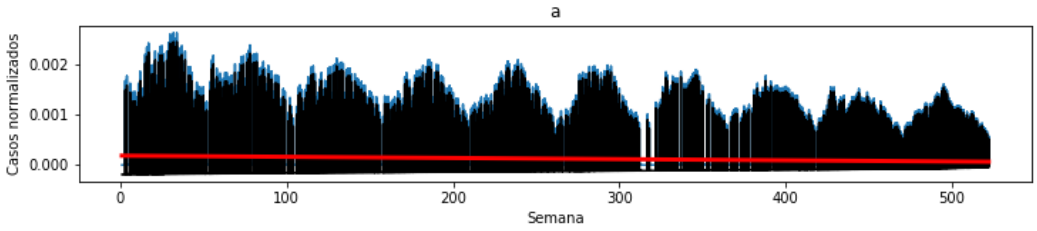


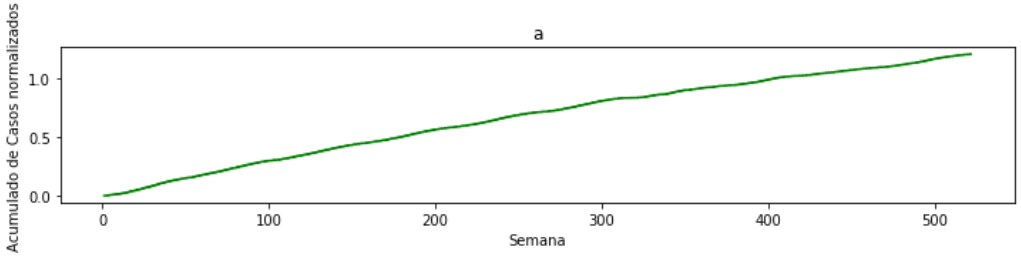




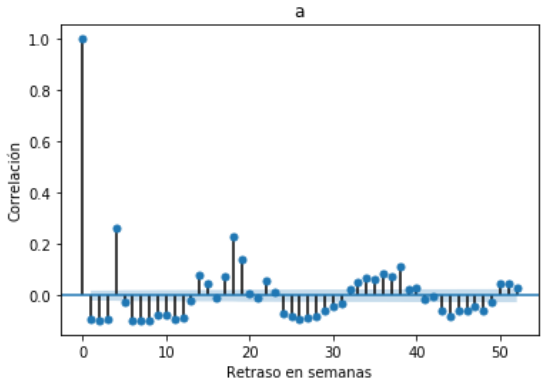

```
for name, group in cieG:

    # https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.detrend.html
    detrended = signal.detrend(group.casos)

    a, b, r, p, e = stats.linregress(group['sem'], group.casos)
    print("y = f(x) = {} x + {}".format(a, b))
    print("error", e)
    print("p = ", p)
    print("pendiente {:s}significativa".format("no " if p >= 0.05 else ""))
    print("R^2", r**2)
    plt.figure(figsize=(12, 2))
    plt.plot(group['sem'], group.casos)
    plt.plot(group['sem'], detrended, c='black')
    plt.plot(group['sem'], (a * group['sem'] + b), label = 'y = {:.1f}x + {:.0f}'.format(a, b), color =
    plt.title(name)
    plt.xlabel("Semana")
    plt.ylabel("Casos normalizados")
    plt.show()

    # https://stackoverflow.com/questions/48497756/time-series-distance-metric
    plt.figure(figsize=(12, 2))
    # https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.Series.cumsum.html
    plt.plot(group['sem'], group.casos.cumsum(), c='green')
    plt.title(name)
    plt.xlabel("Semana")
    plt.ylabel("Acumulado de Casos normalizados")
    plt.show()

    plt.figure()
    # https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/
    plot_acf(detrended, lags=52)
    # https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.acf.html
    plt.title(name)
    plt.xlabel("Retraso en semanas")
    plt.ylabel('Correlación')
    plt.show()
```

```
y = f(x) = -2.3224356070873255e-07 x + 0.00018642953885186607
error 2.458069138569354e-08
p =  4.2189376399252386e-21
pendiente significativa
R^2 0.008834750473414046
```

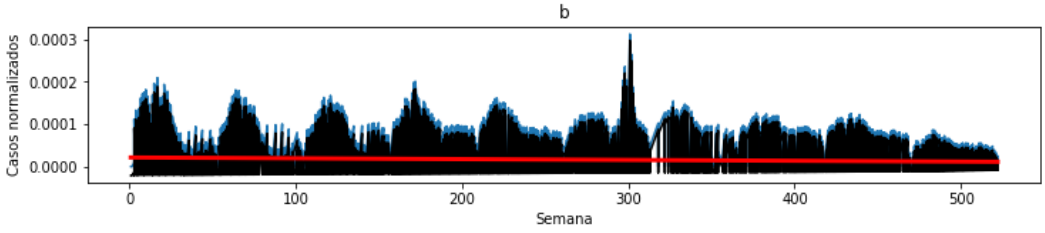


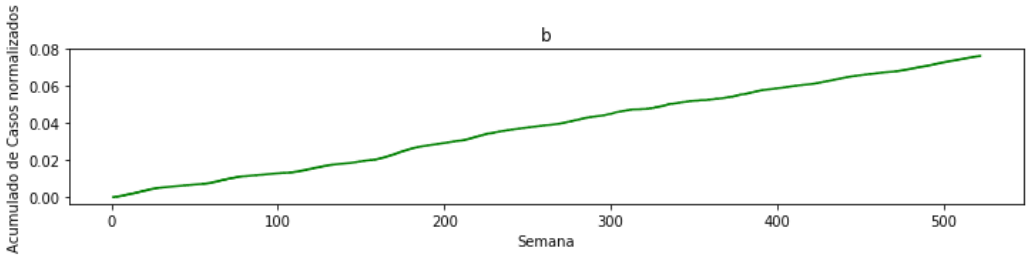



```
<Figure size 432x288 with 0 Axes>
```

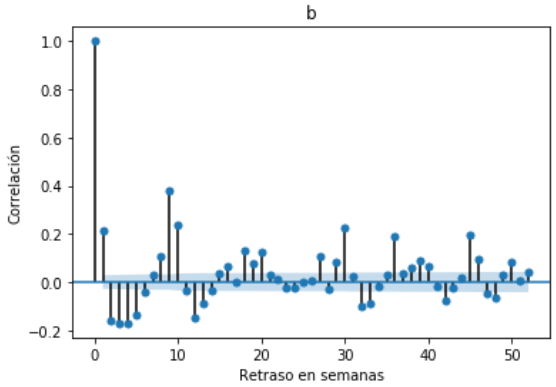



```
y = f(x) = -1.9503026794867653e-08 x + 2.1003475197080275e-05
error 2.9914826904487065e-09
p =  7.747452841909533e-11
pendiente significativa
R^2 0.008415817644051377
```

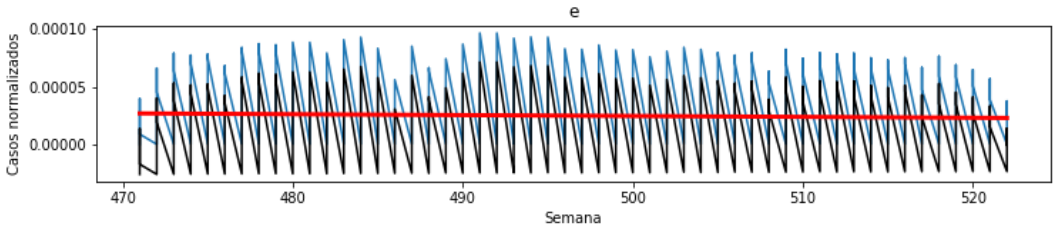


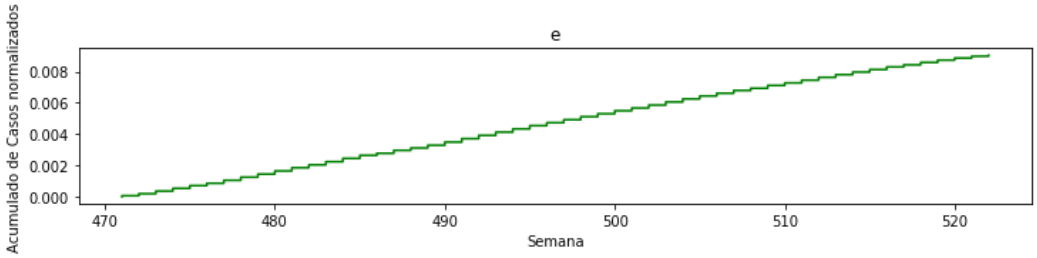



```
<Figure size 432x288 with 0 Axes>
```

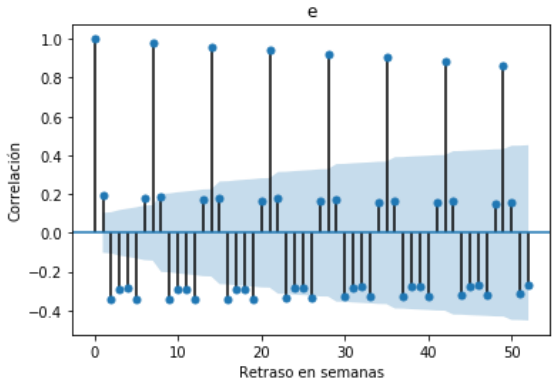



```
y = f(x) = -7.687412126573934e-08 x + 6.301895314999713e-05
error 1.1457310486128478e-07
p =  0.5026730466383222
pendiente no significativa
R^2 0.0012420716051614582
```

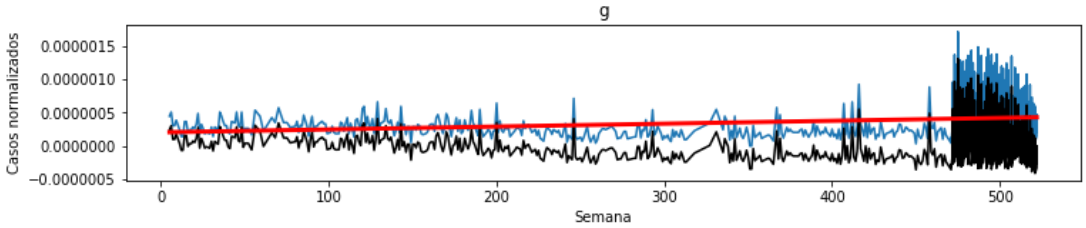


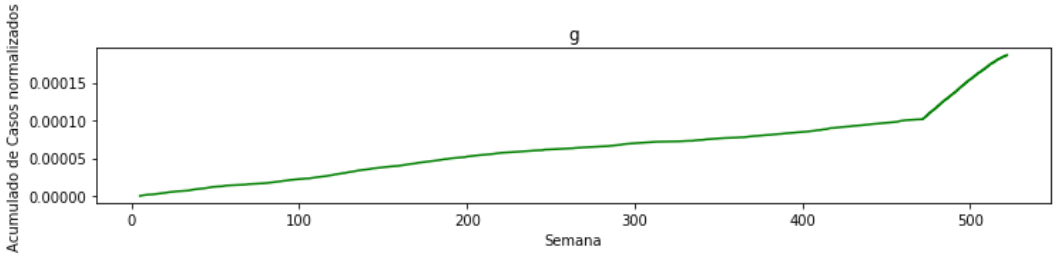



```
<Figure size 432x288 with 0 Axes>
```

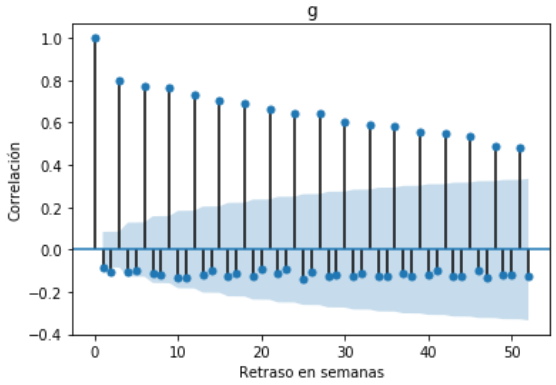



```
y = f(x) = 4.324626671299835e-10 x + 2.04941342972253e-07
error 7.239655510808932e-11
p =  4.1887790791163235e-09
pendiente significativa
R^2 0.06134442012808605
```

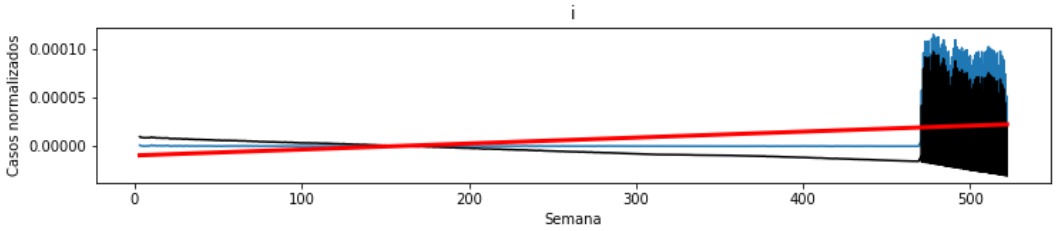


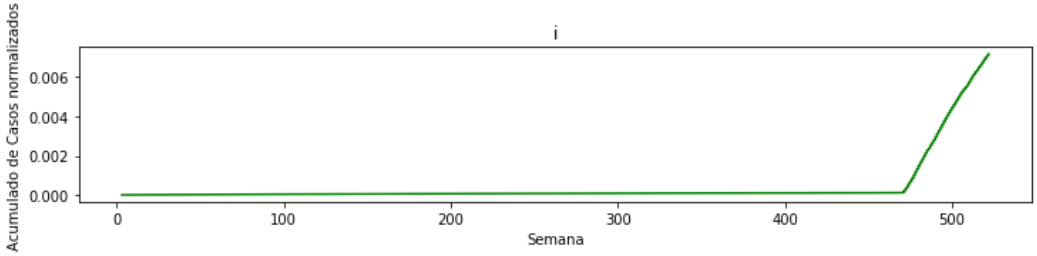



```
<Figure size 432x288 with 0 Axes>
```

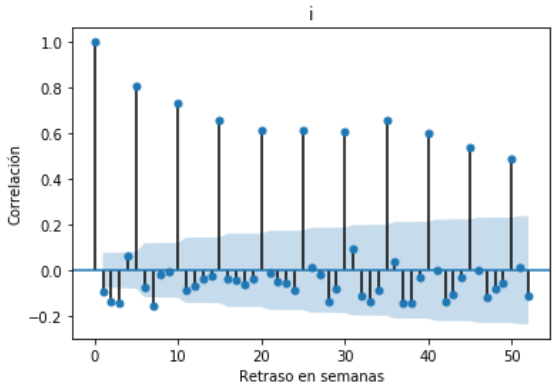


```
y = f(x) = 6.114012516943938e-08 x + -9.37021948204884e-06
error 5.454249521195378e-09
p =  7.943541732517958e-27
pendiente significativa
R^2 0.1593291079485521
```

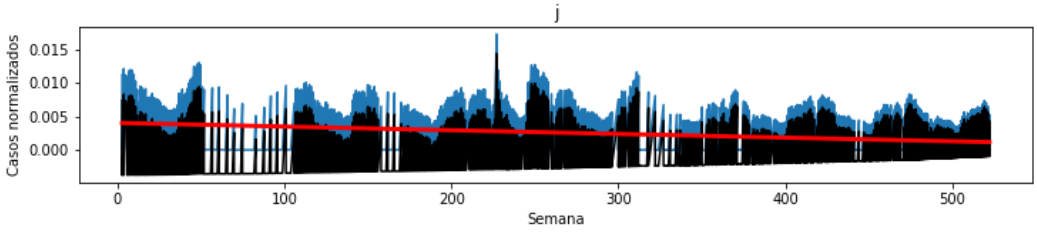


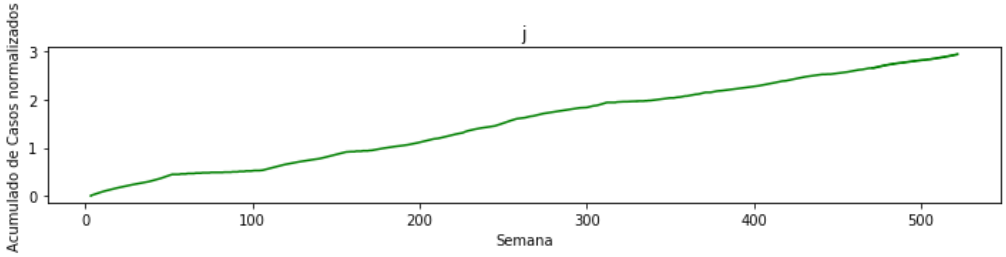



```
<Figure size 432x288 with 0 Axes>
```

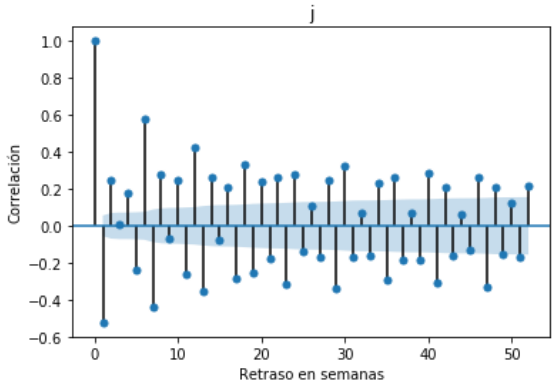



```
y = f(x) = -5.4762876304630354e-06 x + 0.004047543918036931
error 6.313606290646645e-07
p =  1.2891385768087959e-17
pendiente significativa
R^2 0.057202405083796086
```

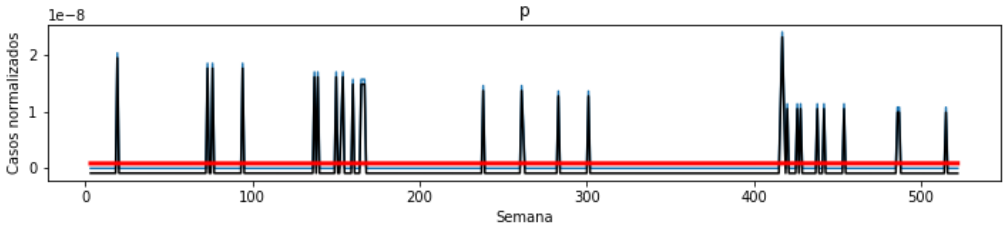


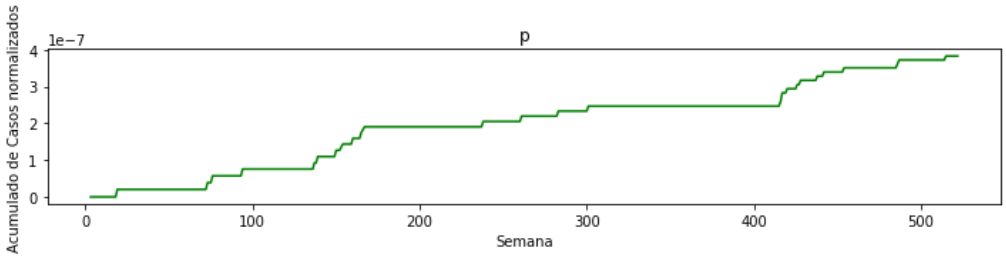



```
<Figure size 432x288 with 0 Axes>
```

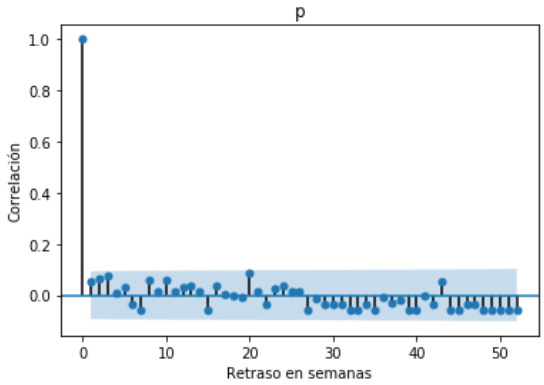



```
y = f(x) = 3.0607452607643954e-14 x + 8.632235917582137e-10
error 1.081125720208934e-12
p =  0.977427221531179
pendiente no significativa
R^2 1.8298987789071008e-06
```

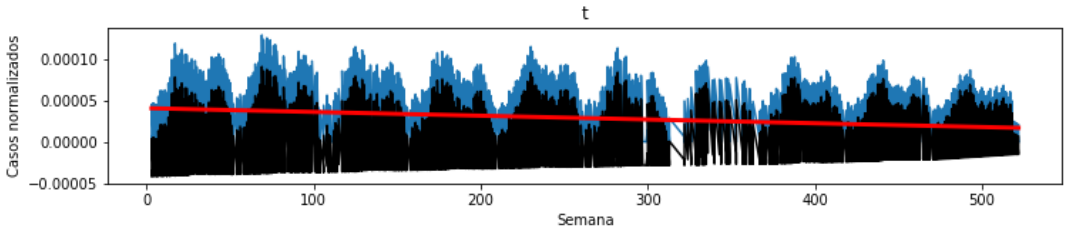


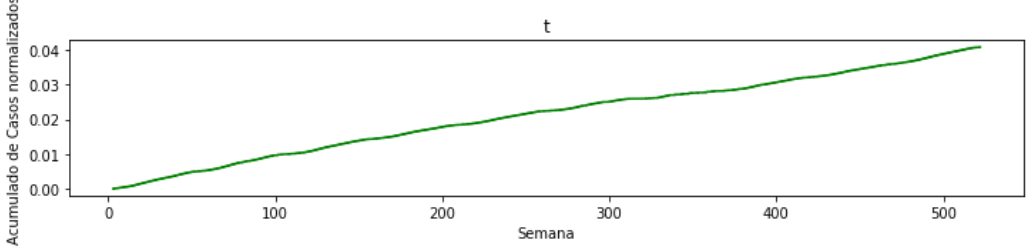



```
<Figure size 432x288 with 0 Axes>
```

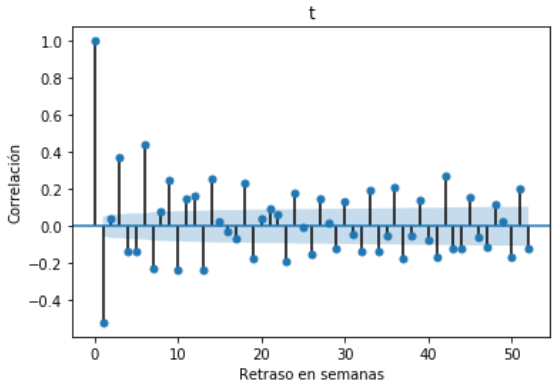



```
y = f(x) = -4.552145867367468e-08 x + 4.150055938885172e-05
error 5.28039387715845e-09
p =  1.7027198403366345e-17
pendiente significativa
R^2 0.04850098103773122
```

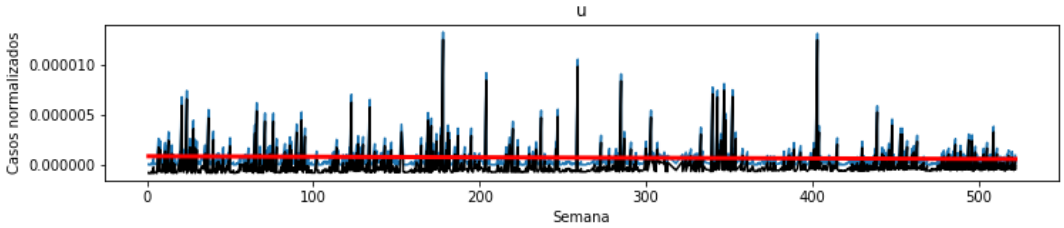


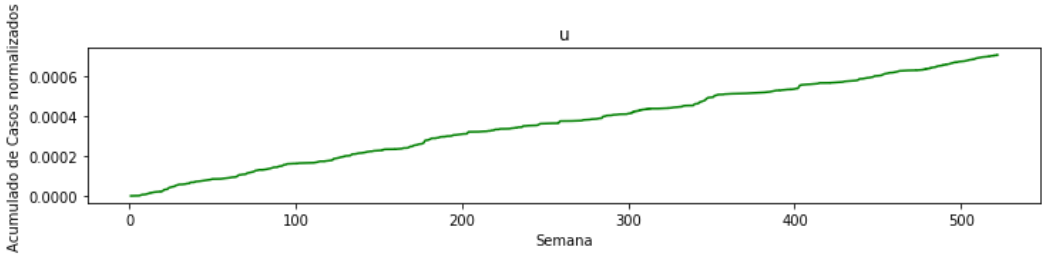



```
<Figure size 432x288 with 0 Axes>
```

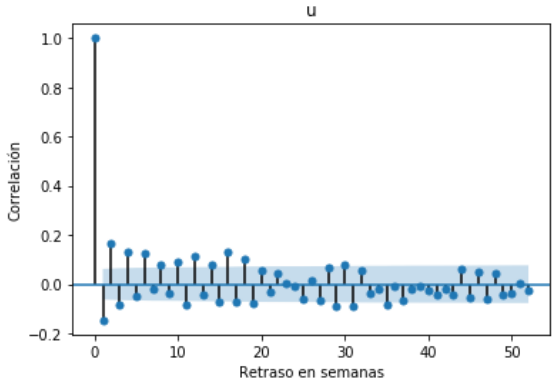



```
y = f(x) = -5.496981234999186e-10 x + 8.572895488021172e-07
error 2.657552949170129e-10
p =  0.038856309457124164
pendiente significativa
R^2 0.004272969521096534
```

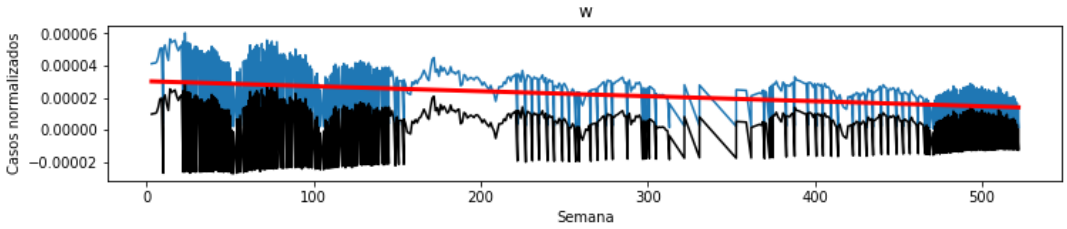


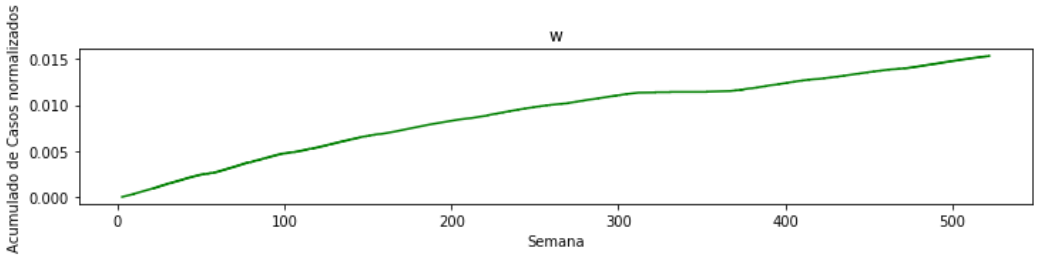



```
<Figure size 432x288 with 0 Axes>
```

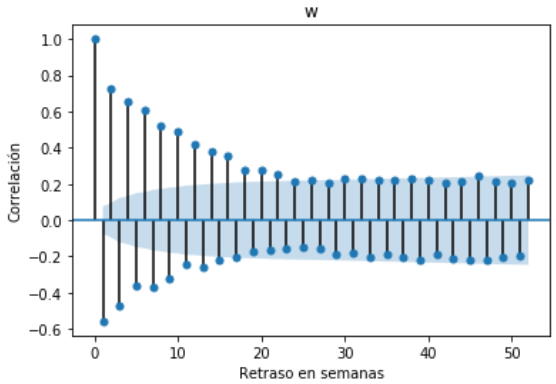


```
y = f(x) = -3.11383856620073e-08 x + 3.0350821580836582e-05
error 3.4613396780786333e-09
p =  2.313921331364859e-18
pendiente significativa
R^2 0.10649538120403663
```

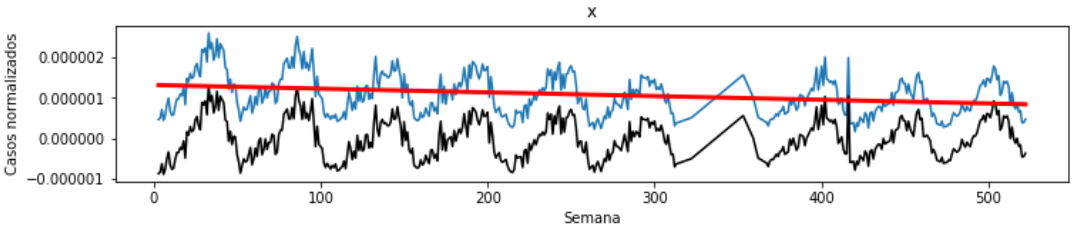


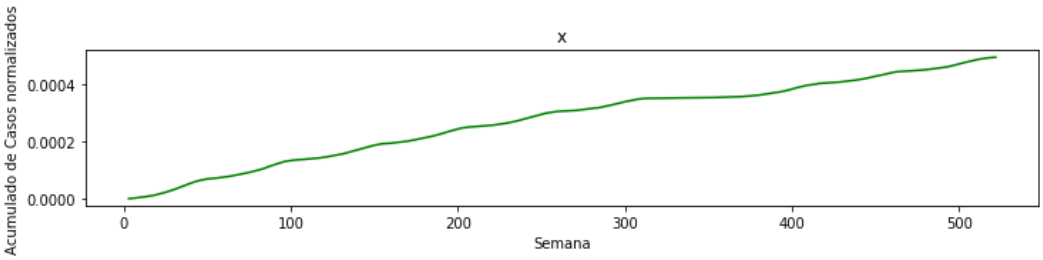



```
<Figure size 432x288 with 0 Axes>
```

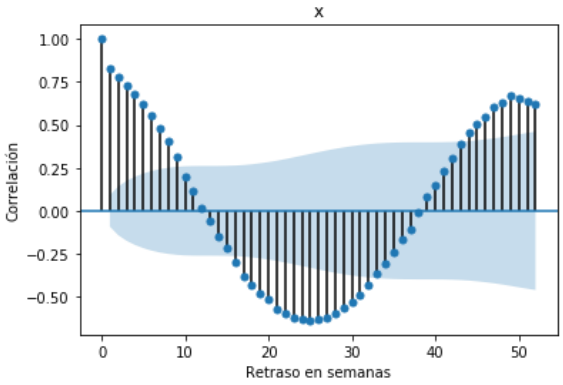



```
y = f(x) = -9.132331371853938e-10 x + 1.3202325787207294e-06
error 1.431149104840528e-10
p =  4.359158383428425e-10
pendiente significativa
R^2 0.08264080526994147
```

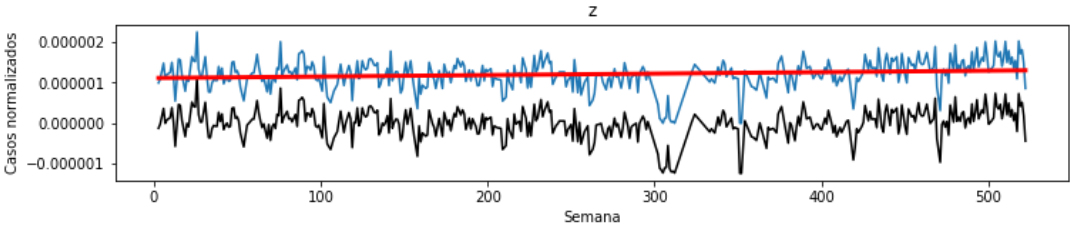


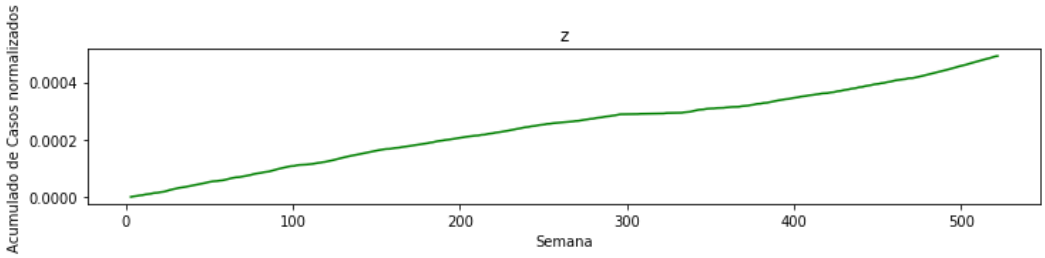



```
<Figure size 432x288 with 0 Axes>
```

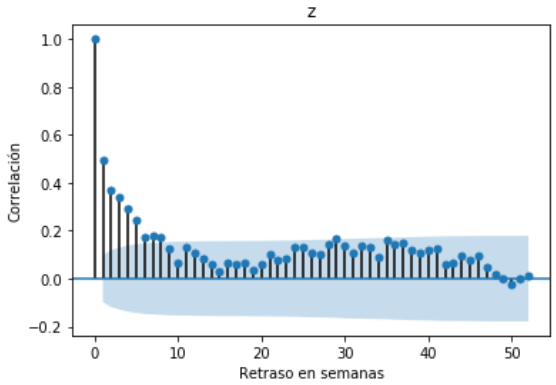



```
y = f(x) = 3.638834463785146e-10 x + 1.1174278631513553e-06
error 1.1201987356227443e-10
p =  0.001257639893824037
pendiente significativa
R^2 0.025453964627936122
```






```
<Figure size 432x288 with 0 Axes>
```



Se extraen las características de cada CIE en tanto serie de tiempo. A saber, su pendiente, ordenada en el origen y las autocorrelaciones con retraso de 1 a 52 semanas (eliminando el retraso de 0 semanas)

```
ciesF = pd.DataFrame(ciesF)

# https://stackoverflow.com/a/11346337
colNames = ['m', 'b']
for i in range (53):
    colNames.append('ac' + str(i))
colNames.append('cie')
ciesF.columns = colNames

ciesF = ciesF.drop(['ac0'], axis=1)

ciesF.sort_values(by=['m'], ascending=False)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	m	b	ac1	ac2	ac3	ac4	ac5	ac6
	3.638834e-	1.117428e-						

38	10	06	0.497210	0.371337	0.338770	0.295026	0.246889	0.17079
11	2.725401e-10	1.947195e-08	0.089659	-0.019229	-0.025867	-0.002608	-0.028985	-0.0011
20	4.306189e-11	4.531178e-07	0.299805	0.225857	0.068495	0.057942	0.001655	0.01753
0	4.268750e-11	-5.612765e-09	0.393749	0.122484	0.010328	0.026661	0.015698	0.02114
8	3.893878e-11	4.432440e-08	0.393243	0.260146	0.044702	0.032172	0.012605	0.02516
9	2.343471e-13	-3.600315e-11	-0.007522	-0.007561	-0.007600	-0.007639	-0.007678	-0.0077
30	3.060745e-14	8.632236e-10	0.055952	0.062823	0.076371	0.006623	0.030863	-0.0372
24	-1.067385e-11	4.355251e-09	0.077117	0.322187	0.046591	0.036037	-0.023600	0.02012
6	-1.462773e-11	4.515318e-08	0.085993	0.048897	0.064574	0.118768	-0.023565	-0.0264
35	-2.343882e-11	1.596244e-07	0.550097	0.512561	0.454005	0.411662	0.363330	0.2544
10	-3.423410e-11	1.862211e-08	-0.081445	0.057375	-0.049496	0.012841	0.056102	-0.0202
14	-4.514430e-11	5.003110e-08	0.076912	0.149122	0.088945	0.072881	0.114257	0.07408
13	-1.074456e-10	5.326971e-08	0.003941	-0.012528	-0.008077	-0.013007	-0.007537	-0.0164
19	-1.241134e-10	2.116017e-07	0.289203	0.226359	0.276856	0.274657	0.253692	0.24741
7	-1.764268e-10	6.894953e-07	0.522652	0.363609	0.361372	0.385651	0.326709	0.3255
26	-2.654424e-10	3.204952e-07	0.110847	0.069672	0.107836	0.080559	0.128797	-0.0270
17	-2.722126e-10	1.262294e-07	0.488103	0.474707	0.426456	0.414057	0.317635	0.3042
27	-4.556172e-10	3.750011e-07	0.232941	0.165279	0.155966	0.155952	0.159776	0.11007
31	-7.142618e-10	1.178303e-06	0.791010	0.750958	0.691090	0.610929	0.512450	0.43414
34	-8.130422e-10	1.482782e-06	0.026306	0.067629	0.068971	0.106583	0.028512	0.00752
	-9.132331e-	1.320233e-						

37	10	06	0.826727	0.781848	0.727522	0.675235	0.624828	0.5509
21	-9.851453e-10	1.437287e-06	0.149215	0.272471	0.152501	0.089044	0.100963	0.1068
25	-1.183463e-09	7.100912e-07	0.388003	0.416624	0.313638	0.387392	0.284052	0.2503
15	-2.875762e-09	8.658057e-06	0.895262	0.842421	0.792705	0.707147	0.622664	0.5277
2	-3.075234e-09	1.290497e-05	0.667701	0.505751	0.436771	0.352219	0.289719	0.2284
5	-3.831012e-09	4.559111e-06	0.380632	0.195406	0.107774	0.091505	0.073293	0.0731
12	-4.255464e-09	2.747090e-06	0.140500	0.121196	0.127669	0.099854	0.073198	0.0272
22	-4.597787e-09	2.970356e-06	0.564047	0.398251	0.335638	0.284887	0.179178	0.0788
18	-7.433435e-09	6.901512e-06	0.765747	0.587239	0.516973	0.418962	0.348350	0.2393
4	-9.213574e-09	1.295795e-05	0.605797	0.479571	0.410353	0.367218	0.341321	0.3113
3	-1.078567e-08	6.155839e-06	0.620012	0.595386	0.535631	0.462663	0.438088	0.3731
32	-1.315899e-08	1.643445e-05	0.903920	0.826842	0.732156	0.614444	0.497397	0.3768
23	-3.141197e-08	1.115938e-04	0.767118	0.547797	0.395471	0.291686	0.210417	0.1284
29	-4.387659e-08	5.089286e-05	0.927733	0.858740	0.790416	0.727838	0.637424	0.5450
33	-5.001284e-08	8.796233e-05	0.881185	0.793629	0.694008	0.596382	0.456681	0.3192
36	-5.193298e-08	4.533358e-05	0.844756	0.768593	0.723830	0.666654	0.614451	0.5645
16	-1.464468e-07	1.121105e-04	0.946239	0.892301	0.837622	0.778140	0.711102	0.6292
1	-1.765308e-06	1.975491e-03	0.831565	0.690654	0.594390	0.516446	0.437316	0.3268
28	-6.932315e-06	8.805968e-03	0.853479	0.727355	0.620092	0.535945	0.445607	0.3496

39 rows × 55 columns

Se agrupan las CIEs por casos dados en una semana, como series de tiempo.

```
ciesTS = pd.DataFrame(ciesTSt)
t = list(cie.groups.keys())
t.remove('a92.3')
ciesTS['cie'] = t

ciesTS.sample(3)
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	0	1	2	3	4	5	6	
12	3.413076e-06	2.295700e-06	2.031593e-06	2.620755e-06	3.331812e-06	3.372444e-06	3.331812e-06	4.3 06
20	4.875822e-07	5.078982e-07	4.063185e-07	4.266345e-07	4.266345e-07	6.297937e-07	4.875822e-07	6.5 07
25	7.110574e-07	7.110574e-07	1.015796e-07	9.548485e-07	1.422115e-07	3.047389e-07	2.031593e-07	2.6 07

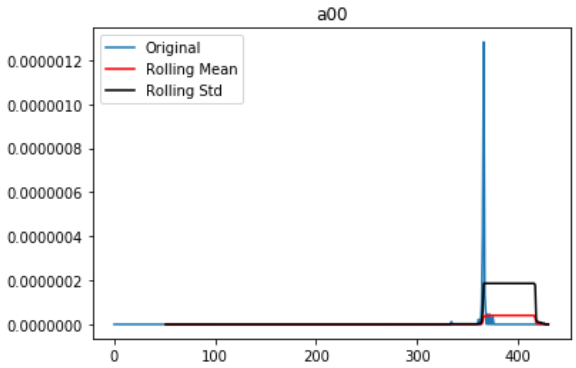
3 rows × 509 columns

Pronóstico

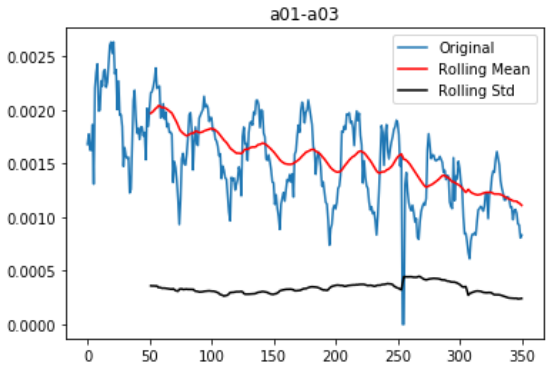
Una vez eliminada la tendencia se puede comprobar que las series de tiempo para cada enfermedad son estacionales

```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries, w, name):

    #Determing rolling statistics
    rolmean = timeseries.rolling(w).mean()
    rolstd = timeseries.rolling(w).std()

    #Plot rolling statistics:
    orig = plt.plot(timeseries,label='Original')
    mean = plt.plot(rolmean, color='red', label='Rolling Mean')
    std = plt.plot(rolstd, color='black', label = 'Rolling Std')
    plt.legend(loc='best')
    plt.title(name)
    plt.show(block=False)

    #Perform Dickey-Fuller test:
    print('Results of Dickey-Fuller Test:')
    dftest = adfuller(timeseries, autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','#Lags Used','Number of Observa
```

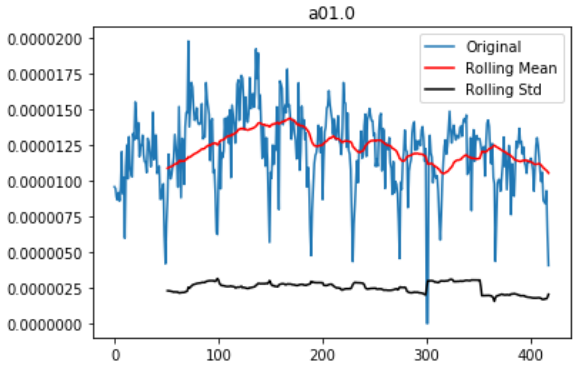
```
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)

for i in range(len(ciesTS)):
    temp = ciesTS.iloc[i, : 508]
    temp.dropna(inplace=True)
    test_stationarity(temp, 52, ciesTS.iloc[i, -1])
```



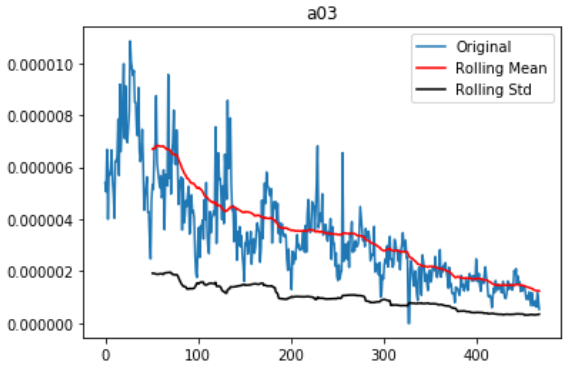

```
Results of Dickey-Fuller Test:
Test Statistic                -1.356054e+01
p-value                        2.314720e-25
#Lags Used                     0.000000e+00
Number of Observations Used    4.300000e+02
Critical Value (1%)           -3.445649e+00
Critical Value (5%)           -2.868285e+00
Critical Value (10%)          -2.570363e+00
dtype: float64
```



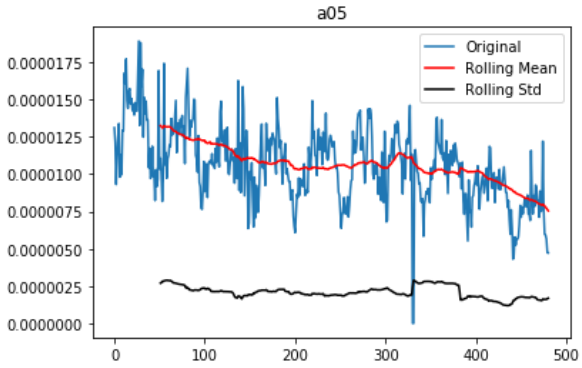

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.424746
p-value                          0.010144
#Lags Used                       2.000000
Number of Observations Used    348.000000
Critical Value (1%)             -3.449282
Critical Value (5%)             -2.869881
Critical Value (10%)            -2.571214
dtype: float64
```



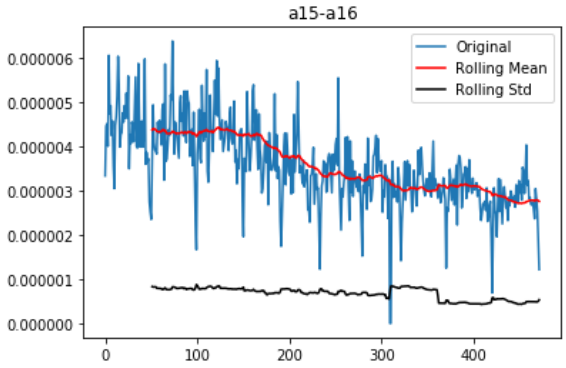

```
Results of Dickey-Fuller Test:
Test Statistic                -5.719534e+00
p-value                        6.993272e-07
#Lags Used                     2.000000e+00
Number of Observations Used    4.150000e+02
Critical Value (1%)           -3.446206e+00
Critical Value (5%)           -2.868530e+00
Critical Value (10%)          -2.570493e+00
dtype: float64
```



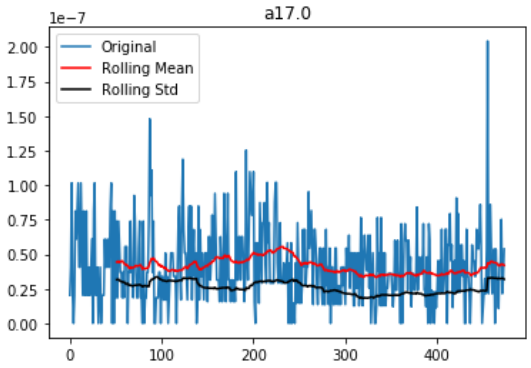

```
Results of Dickey-Fuller Test:
Test Statistic                  -2.106851
p-value                          0.241769
#Lags Used                       7.000000
Number of Observations Used    461.000000
Critical Value (1%)             -3.444615
Critical Value (5%)             -2.867830
Critical Value (10%)            -2.570120
dtype: float64
```



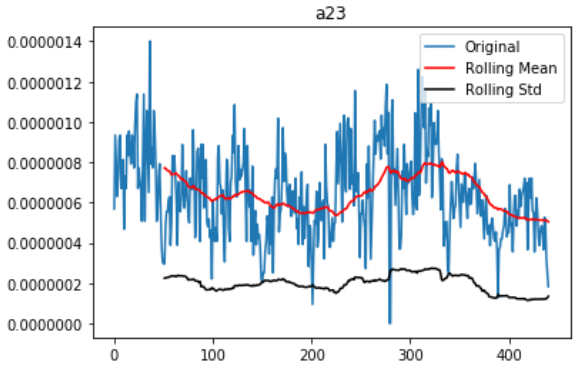

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.760647
p-value                          0.003337
#Lags Used                       4.000000
Number of Observations Used    476.000000
Critical Value (1%)             -3.444163
Critical Value (5%)             -2.867631
Critical Value (10%)            -2.570014
dtype: float64
```



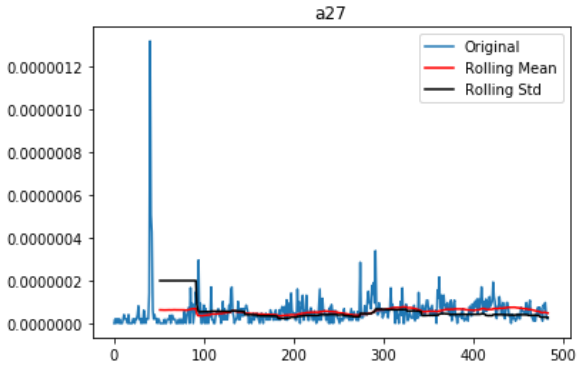

```
Results of Dickey-Fuller Test:
Test Statistic                  -2.133072
p-value                          0.231426
#Lags Used                      10.000000
Number of Observations Used    461.000000
Critical Value (1%)             -3.444615
Critical Value (5%)             -2.867830
Critical Value (10%)            -2.570120
dtype: float64
```



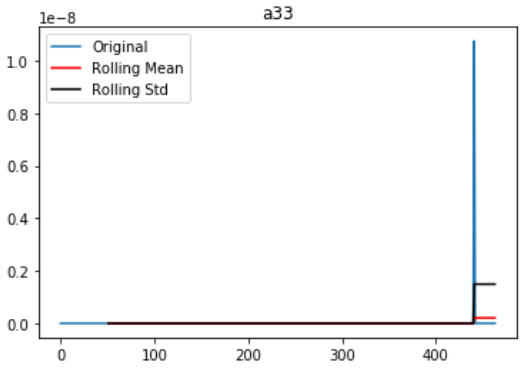

```
Results of Dickey-Fuller Test:
Test Statistic                -8.720275e+00
p-value                        3.400303e-14
#Lags Used                     3.000000e+00
Number of Observations Used    4.700000e+02
Critical Value (1%)           -3.444340e+00
Critical Value (5%)           -2.867709e+00
Critical Value (10%)          -2.570056e+00
dtype: float64
```



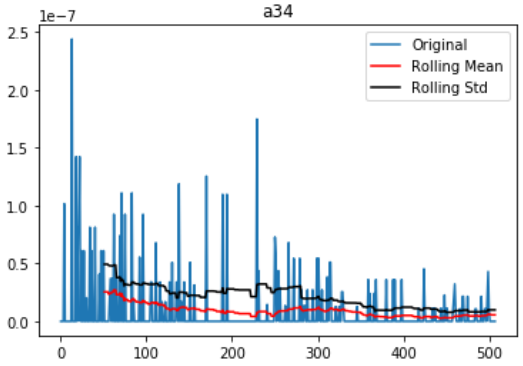

```
Results of Dickey-Fuller Test:
Test Statistic                  -4.078834
p-value                          0.001050
#Lags Used                       5.000000
Number of Observations Used    435.000000
Critical Value (1%)             -3.445473
Critical Value (5%)             -2.868207
Critical Value (10%)            -2.570321
dtype: float64
```



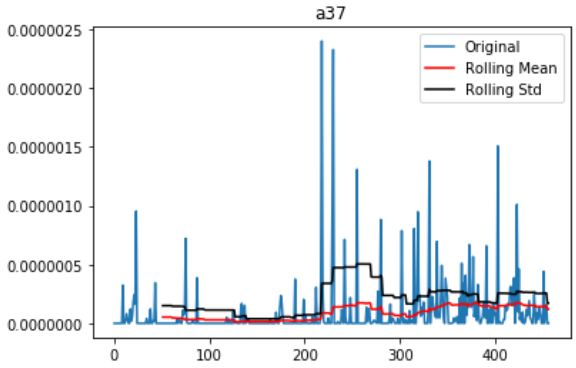

```
Results of Dickey-Fuller Test:
Test Statistic                -1.063365e+01
p-value                        5.137862e-19
#Lags Used                     2.000000e+00
Number of Observations Used    4.810000e+02
Critical Value (1%)           -3.444018e+00
Critical Value (5%)           -2.867568e+00
Critical Value (10%)          -2.569980e+00
dtype: float64
```



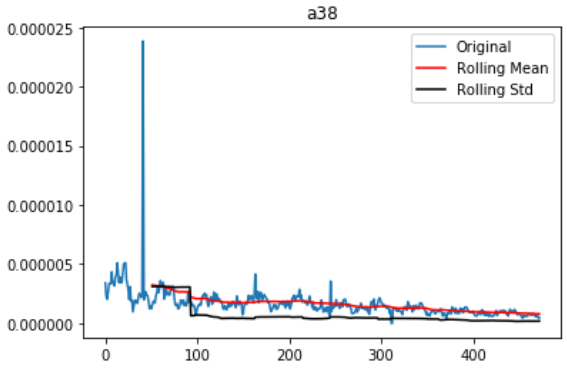

```
Results of Dickey-Fuller Test:
Test Statistic                -21.517435
p-value                         0.000000
#Lags Used                      0.000000
Number of Observations Used    463.000000
Critical Value (1%)             -3.444553
Critical Value (5%)             -2.867803
Critical Value (10%)            -2.570106
dtype: float64
```



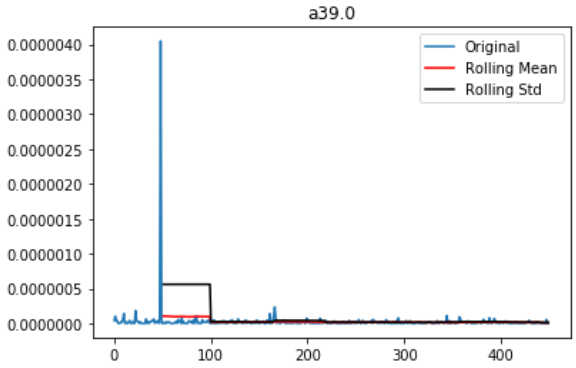

```
Results of Dickey-Fuller Test:
Test Statistic                -1.468685e+01
p-value                        3.093388e-27
#Lags Used                     1.000000e+00
Number of Observations Used    5.060000e+02
Critical Value (1%)           -3.443340e+00
Critical Value (5%)           -2.867269e+00
Critical Value (10%)          -2.569821e+00
dtype: float64
```



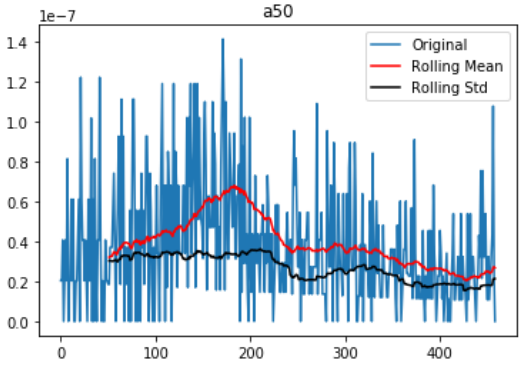

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.809081
p-value                          0.002816
#Lags Used                      11.000000
Number of Observations Used    445.000000
Critical Value (1%)             -3.445131
Critical Value (5%)             -2.868057
Critical Value (10%)            -2.570241
dtype: float64
```



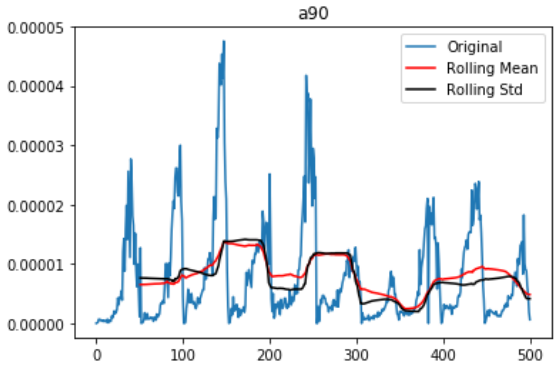

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.053787
p-value                          0.000017
#Lags Used                       4.000000
Number of Observations Used    468.000000
Critical Value (1%)             -3.444400
Critical Value (5%)             -2.867736
Critical Value (10%)            -2.570070
dtype: float64
```



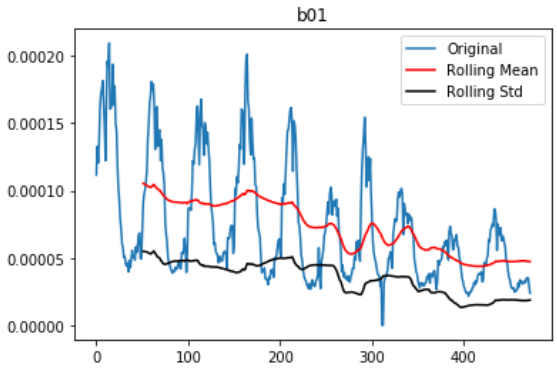

```
Results of Dickey-Fuller Test:
Test Statistic                -20.908062
p-value                         0.000000
#Lags Used                      0.000000
Number of Observations Used    449.000000
Critical Value (1%)             -3.444998
Critical Value (5%)             -2.867999
Critical Value (10%)            -2.570210
dtype: float64
```



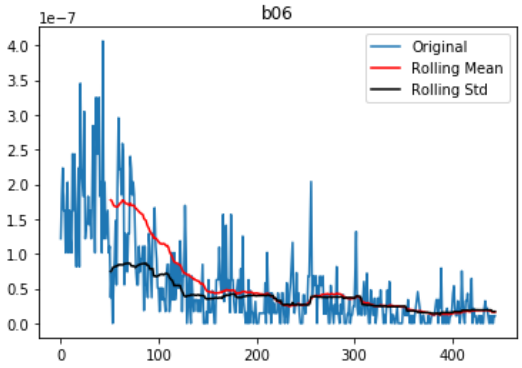

```
Results of Dickey-Fuller Test:
Test Statistic                  -2.973026
p-value                          0.037491
#Lags Used                      11.000000
Number of Observations Used    447.000000
Critical Value (1%)             -3.445064
Critical Value (5%)             -2.868028
Critical Value (10%)            -2.570226
dtype: float64
```



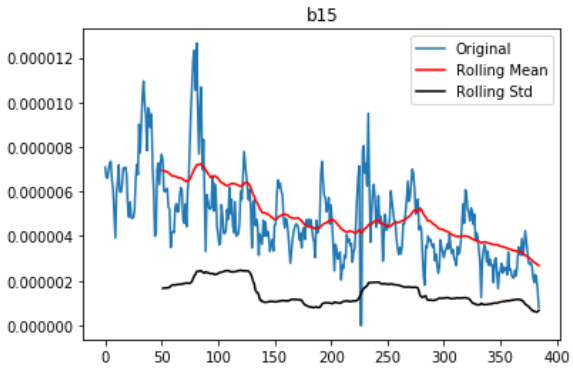

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.304842
p-value                          0.000005
#Lags Used                      12.000000
Number of Observations Used    488.000000
Critical Value (1%)             -3.443821
Critical Value (5%)             -2.867481
Critical Value (10%)            -2.569934
dtype: float64
```



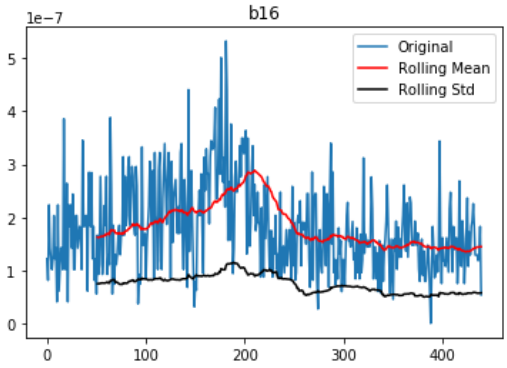

```
Results of Dickey-Fuller Test:
Test Statistic                -6.501620e+00
p-value                        1.158034e-08
#Lags Used                     9.000000e+00
Number of Observations Used    4.630000e+02
Critical Value (1%)           -3.444553e+00
Critical Value (5%)           -2.867803e+00
Critical Value (10%)          -2.570106e+00
dtype: float64
```



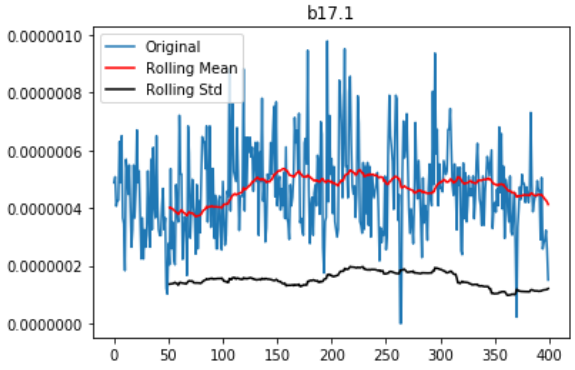

```
Results of Dickey-Fuller Test:
Test Statistic                  -1.996208
p-value                          0.288238
#Lags Used                      11.000000
Number of Observations Used    433.000000
Critical Value (1%)             -3.445543
Critical Value (5%)             -2.868238
Critical Value (10%)            -2.570338
dtype: float64
```



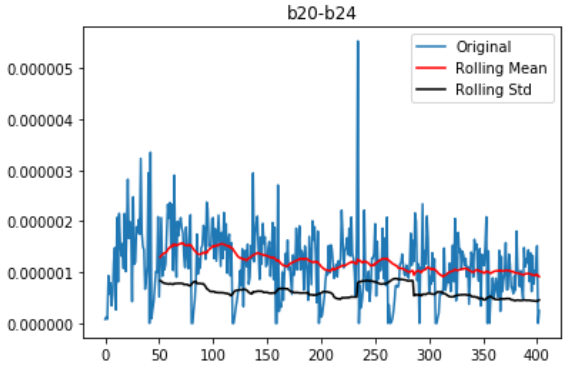

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.403355
p-value                          0.010845
#Lags Used                       8.000000
Number of Observations Used    376.000000
Critical Value (1%)             -3.447862
Critical Value (5%)             -2.869258
Critical Value (10%)            -2.570881
dtype: float64
```



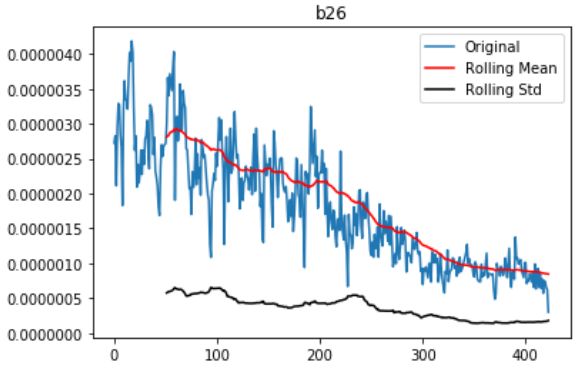

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.818212
p-value                          0.002726
#Lags Used                       6.000000
Number of Observations Used    433.000000
Critical Value (1%)             -3.445543
Critical Value (5%)             -2.868238
Critical Value (10%)            -2.570338
dtype: float64
```



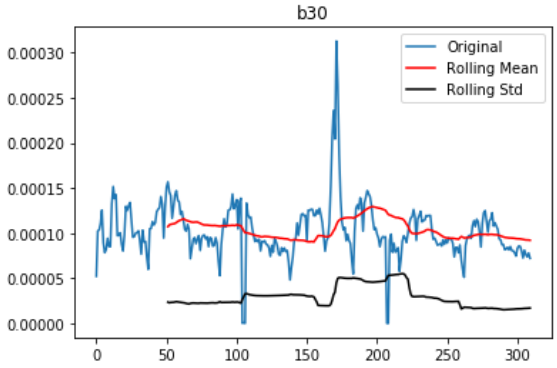

```
Results of Dickey-Fuller Test:
Test Statistic                -9.927614e+00
p-value                        2.878961e-17
#Lags Used                     1.000000e+00
Number of Observations Used    3.980000e+02
Critical Value (1%)           -3.446888e+00
Critical Value (5%)           -2.868829e+00
Critical Value (10%)          -2.570653e+00
dtype: float64
```



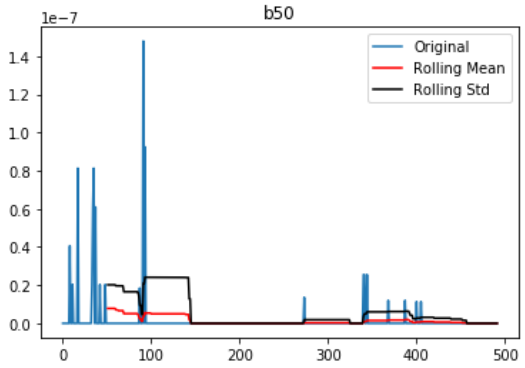

```
Results of Dickey-Fuller Test:
Test Statistic                -7.628780e+00
p-value                        2.034981e-11
#Lags Used                     2.000000e+00
Number of Observations Used    4.000000e+02
Critical Value (1%)           -3.446804e+00
Critical Value (5%)           -2.868793e+00
Critical Value (10%)          -2.570634e+00
dtype: float64
```



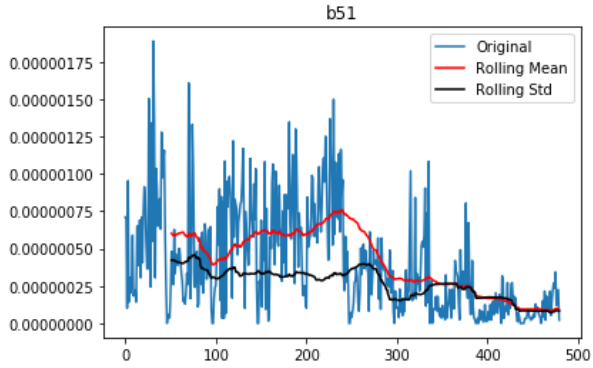

```
Results of Dickey-Fuller Test:
Test Statistic                  -1.383949
p-value                          0.589947
#Lags Used                      14.000000
Number of Observations Used    408.000000
Critical Value (1%)             -3.446480
Critical Value (5%)             -2.868650
Critical Value (10%)            -2.570557
dtype: float64
```



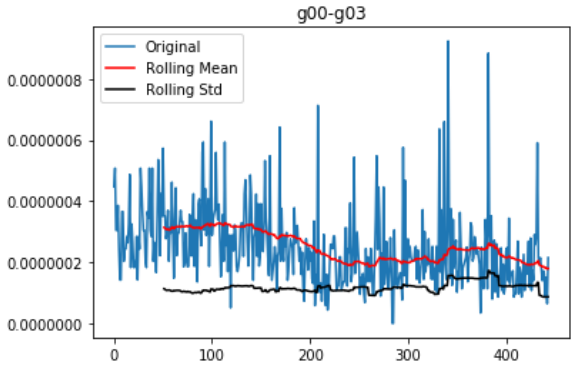

```
Results of Dickey-Fuller Test:
Test Statistic                -6.469065e+00
p-value                        1.381678e-08
#Lags Used                     1.000000e+00
Number of Observations Used    3.080000e+02
Critical Value (1%)           -3.451761e+00
Critical Value (5%)           -2.870970e+00
Critical Value (10%)          -2.571794e+00
dtype: float64
```



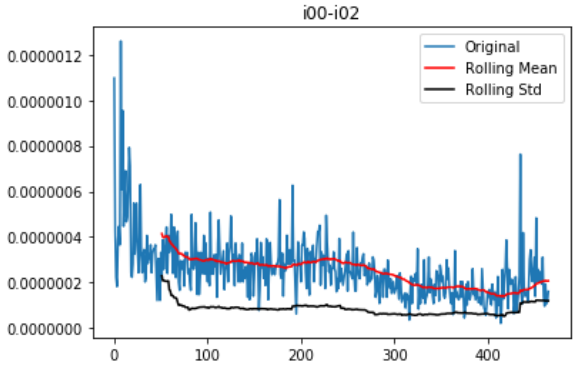

```
Results of Dickey-Fuller Test:
Test Statistic                -1.049502e+01
p-value                        1.121338e-18
#Lags Used                     1.000000e+00
Number of Observations Used    4.900000e+02
Critical Value (1%)           -3.443766e+00
Critical Value (5%)           -2.867457e+00
Critical Value (10%)          -2.569921e+00
dtype: float64
```



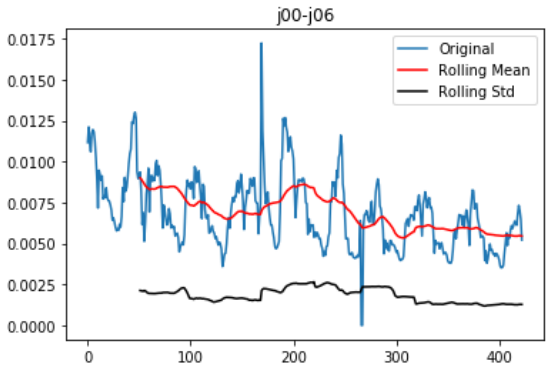

```
Results of Dickey-Fuller Test:
Test Statistic                  -3.050080
p-value                          0.030475
#Lags Used                       8.000000
Number of Observations Used    471.000000
Critical Value (1%)             -3.444310
Critical Value (5%)             -2.867696
Critical Value (10%)            -2.570049
dtype: float64
```



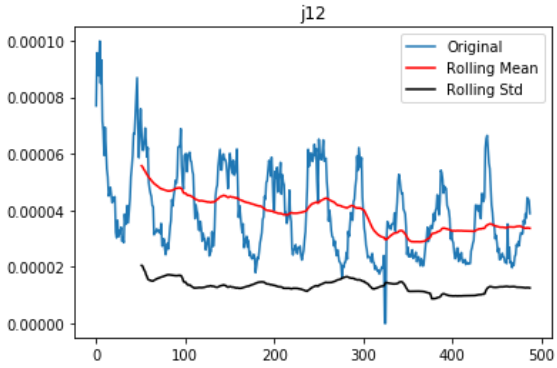

```
Results of Dickey-Fuller Test:
Test Statistic                  -4.194474
p-value                          0.000673
#Lags Used                       8.000000
Number of Observations Used    435.000000
Critical Value (1%)             -3.445473
Critical Value (5%)             -2.868207
Critical Value (10%)            -2.570321
dtype: float64
```



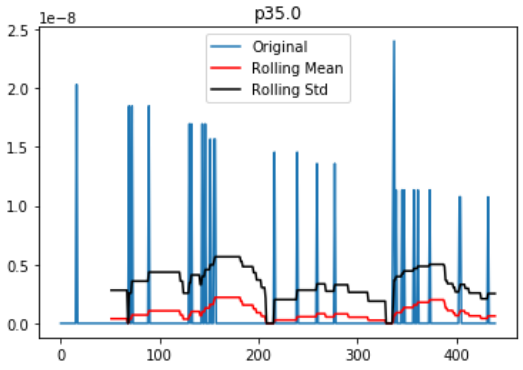

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.300617
p-value                          0.000005
#Lags Used                      17.000000
Number of Observations Used    448.000000
Critical Value (1%)             -3.445031
Critical Value (5%)             -2.868013
Critical Value (10%)            -2.570218
dtype: float64
```



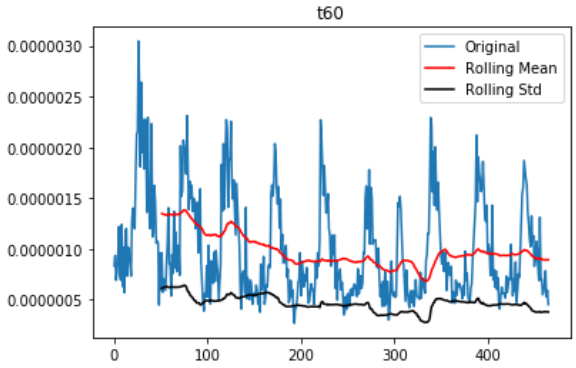

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.196448
p-value                          0.000009
#Lags Used                       0.000000
Number of Observations Used    422.000000
Critical Value (1%)             -3.445941
Critical Value (5%)             -2.868413
Critical Value (10%)            -2.570431
dtype: float64
```



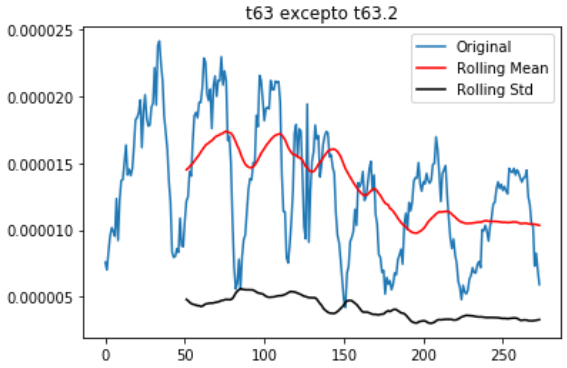

```
Results of Dickey-Fuller Test:
Test Statistic                -6.052569e+00
p-value                        1.267267e-07
#Lags Used                     1.300000e+01
Number of Observations Used    4.740000e+02
Critical Value (1%)           -3.444221e+00
Critical Value (5%)           -2.867657e+00
Critical Value (10%)          -2.570028e+00
dtype: float64
```



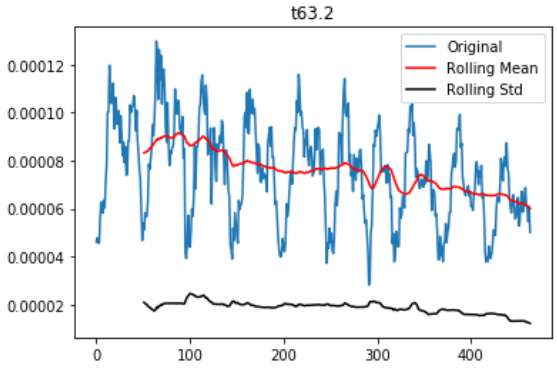

```
Results of Dickey-Fuller Test:
Test Statistic                -1.053776e+01
p-value                        8.810400e-19
#Lags Used                     2.000000e+00
Number of Observations Used    4.370000e+02
Critical Value (1%)           -3.445403e+00
Critical Value (5%)           -2.868177e+00
Critical Value (10%)          -2.570305e+00
dtype: float64
```



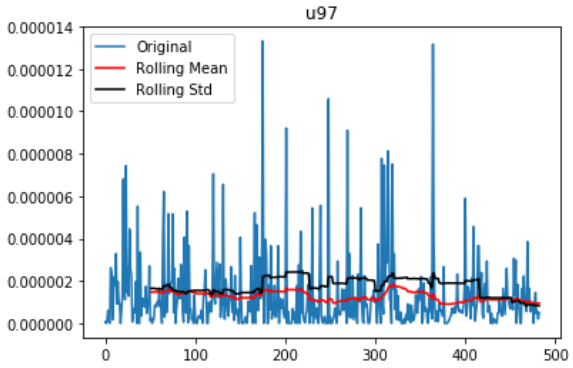

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.316712
p-value                          0.000005
#Lags Used                      14.000000
Number of Observations Used    451.000000
Critical Value (1%)             -3.444933
Critical Value (5%)             -2.867970
Critical Value (10%)            -2.570195
dtype: float64
```



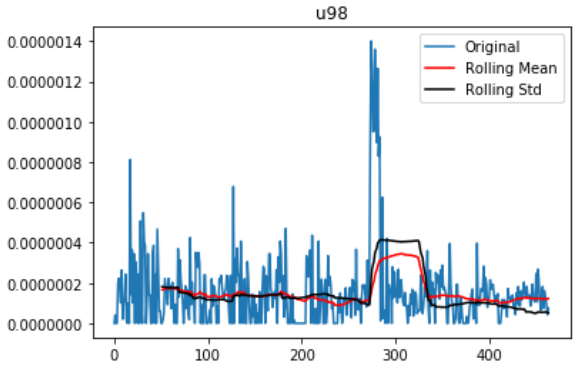

```
Results of Dickey-Fuller Test:
Test Statistic                  -4.113528
p-value                          0.000920
#Lags Used                       4.000000
Number of Observations Used    269.000000
Critical Value (1%)             -3.454896
Critical Value (5%)             -2.872345
Critical Value (10%)            -2.572528
dtype: float64
```



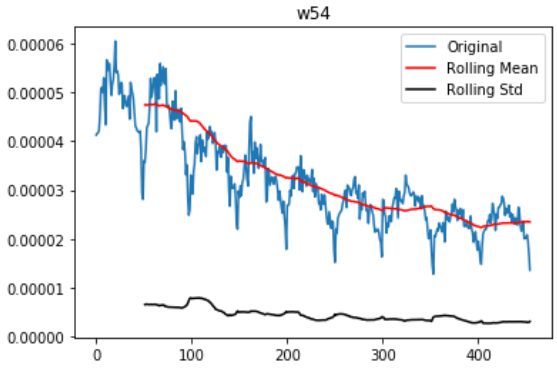

```
Results of Dickey-Fuller Test:
Test Statistic                  -4.874372
p-value                          0.000039
#Lags Used                      11.000000
Number of Observations Used    452.000000
Critical Value (1%)             -3.444900
Critical Value (5%)             -2.867956
Critical Value (10%)            -2.570187
dtype: float64
```



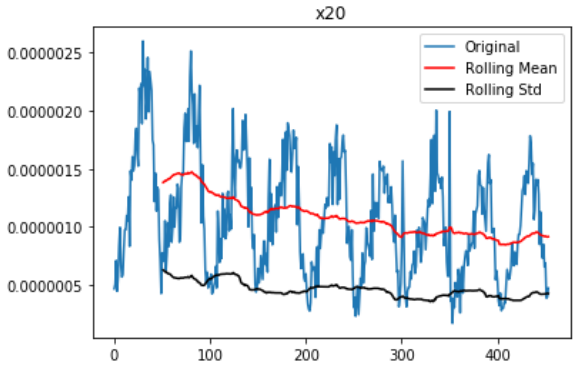

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.375052
p-value                          0.000004
#Lags Used                       8.000000
Number of Observations Used    474.000000
Critical Value (1%)             -3.444221
Critical Value (5%)             -2.867657
Critical Value (10%)            -2.570028
dtype: float64
```



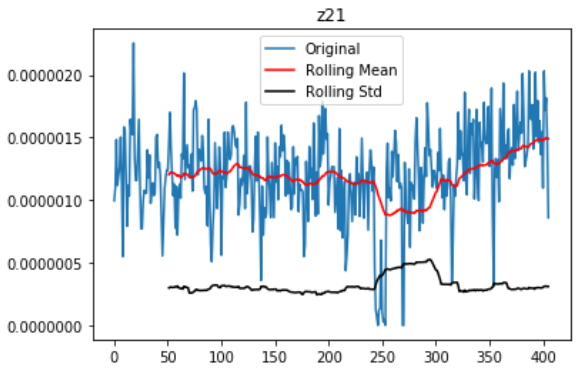

```
Results of Dickey-Fuller Test:
Test Statistic                  -5.342202
p-value                          0.000004
#Lags Used                       5.000000
Number of Observations Used    458.000000
Critical Value (1%)             -3.444709
Critical Value (5%)             -2.867871
Critical Value (10%)            -2.570142
dtype: float64
```




```
Results of Dickey-Fuller Test:
Test Statistic                  -1.813468
p-value                          0.373796
#Lags Used                       2.000000
Number of Observations Used    452.000000
Critical Value (1%)             -3.444900
Critical Value (5%)             -2.867956
Critical Value (10%)            -2.570187
dtype: float64
```



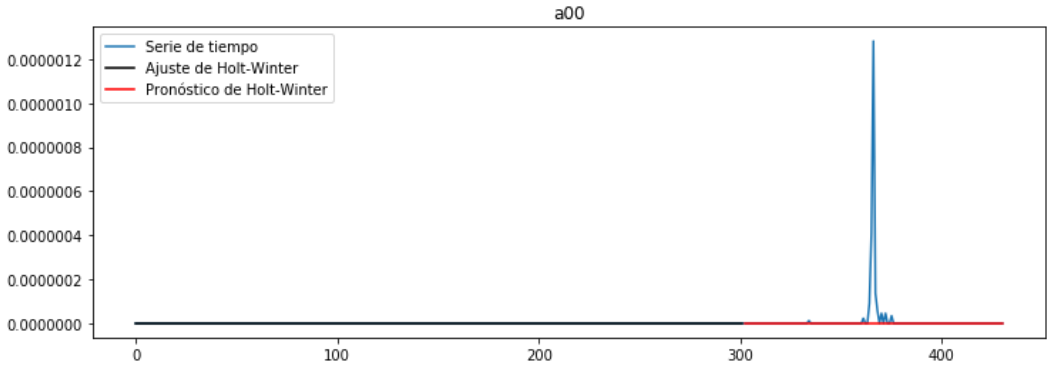

```
Results of Dickey-Fuller Test:
Test Statistic                -6.673468e+00
p-value                        4.526341e-09
#Lags Used                     1.100000e+01
Number of Observations Used    4.420000e+02
Critical Value (1%)           -3.445232e+00
Critical Value (5%)           -2.868101e+00
Critical Value (10%)          -2.570265e+00
dtype: float64
```

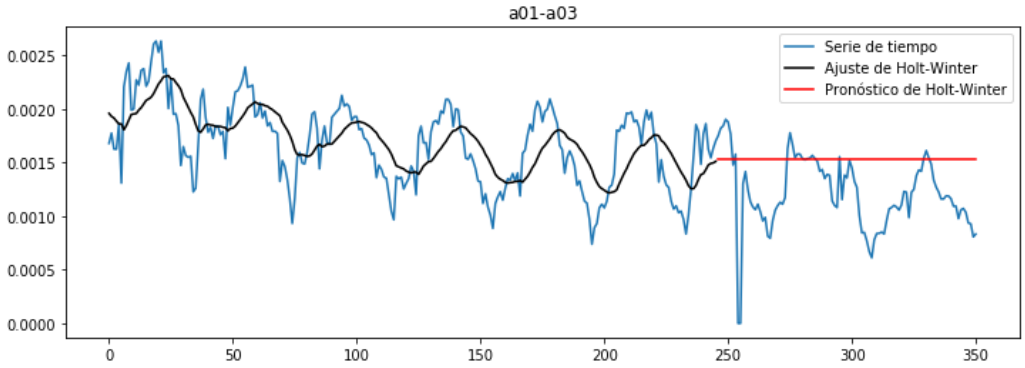


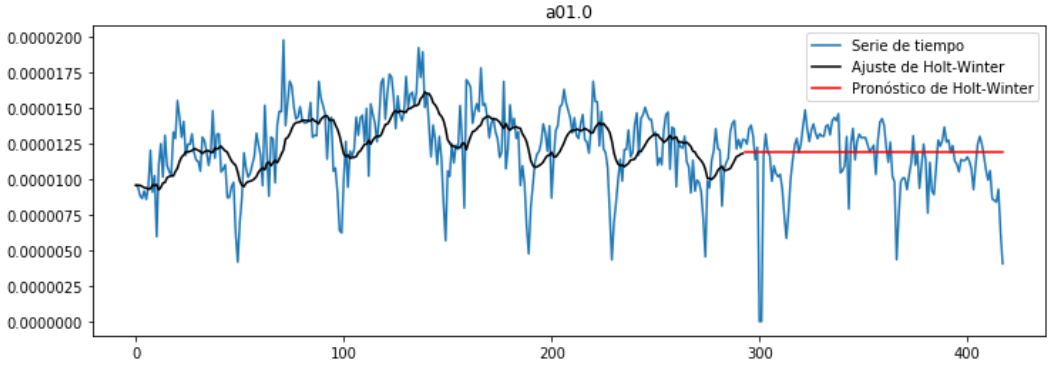

```
Results of Dickey-Fuller Test:
Test Statistic                -6.644475e+00
p-value                        5.308127e-09
#Lags Used                     2.000000e+00
Number of Observations Used    4.030000e+02
Critical Value (1%)           -3.446681e+00
Critical Value (5%)           -2.868739e+00
Critical Value (10%)          -2.570605e+00
dtype: float64
```

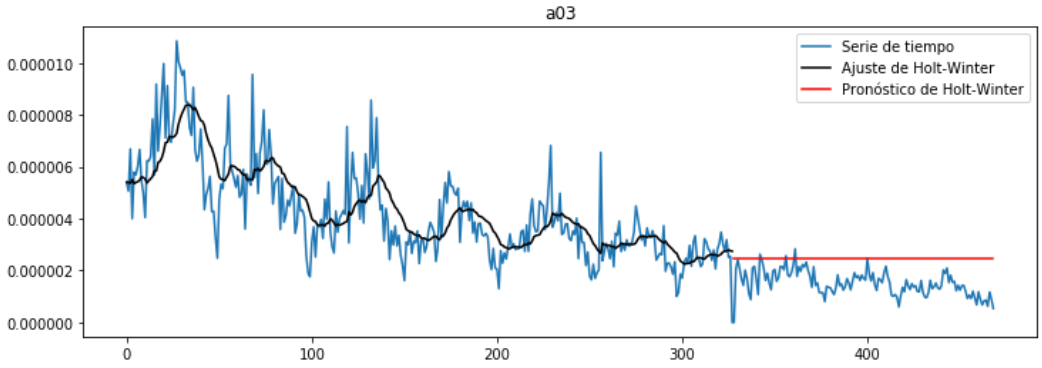
Por ello es posible pronosticarlas con el método de Holt-Winter

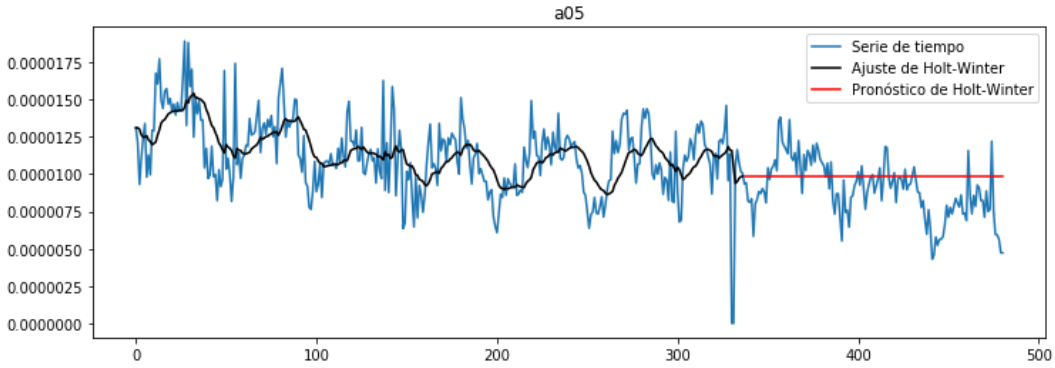
```
# https://www.analyticsvidhya.com/blog/2018/02/time-series-forecasting-methods/
# https://machinelearningmastery.com/time-series-forecasting-methods-in-python-cheat-sheet/
# https://towardsdatascience.com/time-series-in-python-exponential-smoothing-and-arima-processes-2c67f2

from statsmodels.tsa.api import Holt, SimpleExpSmoothing, ExponentialSmoothing

pronosticos = []
real = []
for i in range(len(ciesTS)):
    temp = ciesTS.iloc[i, : 508]
    temp.dropna(inplace=True)

    # https://medium.com/datadriveninvestor/how-to-build-exponential-smoothing-models-using-python-simp
    # Train = 0.7
    train = round(0.7 * len(temp))
    f = ExponentialSmoothing(np.asarray(temp.iloc[0:train])).fit(smoothing_level = 0.1)
    # https://stackoverflow.com/a/50786171
    pred = f.predict(start=train + 1, end=len(temp))
    fcast = f.forecast(len(temp) - train)
    plt.figure(figsize=(12, 4))
    plt.plot(temp)
    plt.plot(f.fittedvalues, c='black')
    plt.plot(range(train, len(temp)), fcast, c='red')
    #plt.plot(range(train, len(temp)), pred, c='red')
    plt.title(ciesTS.iloc[i, 508])
    plt.legend(["Serie de tiempo", "Ajuste de Holt-Winter", "Pronóstico de Holt-Winter"])
    plt.show()


    # https://stackoverflow.com/a/15863028
    real.append(temp.iloc[-1])
    pronosticos.append(fcast[0])
```

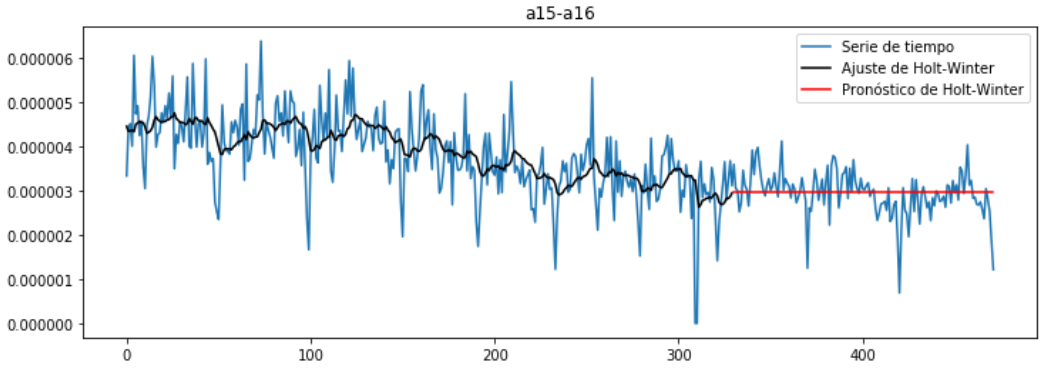

a00
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000012
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
0
100
200
300
400

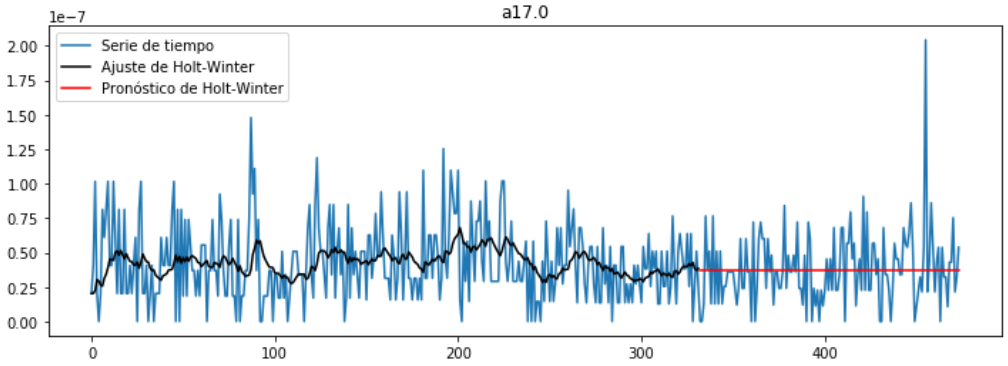

a01-a03
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0025
0.0020
0.0015
0.0010
0.0005
0.0000
0
50
100
150
200
250
300
350

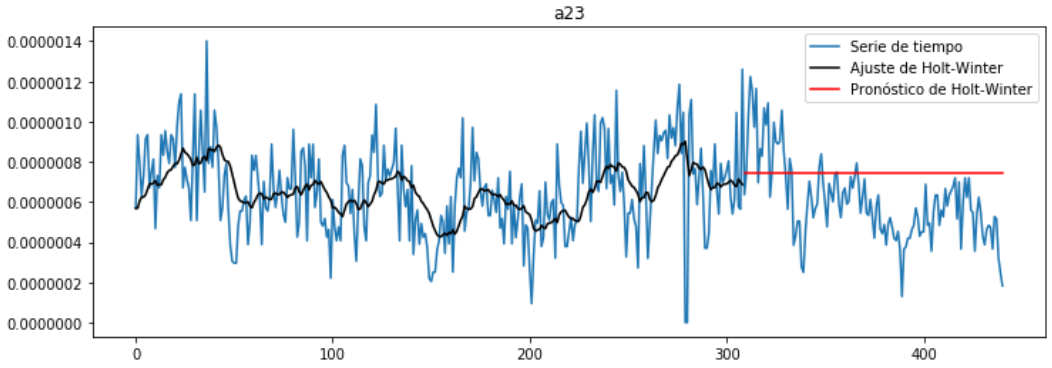

a01.0
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000200
0.0000175
0.0000150
0.0000125
0.0000100
0.0000075
0.0000050
0.0000025
0.0000000
0
100
200
300
400

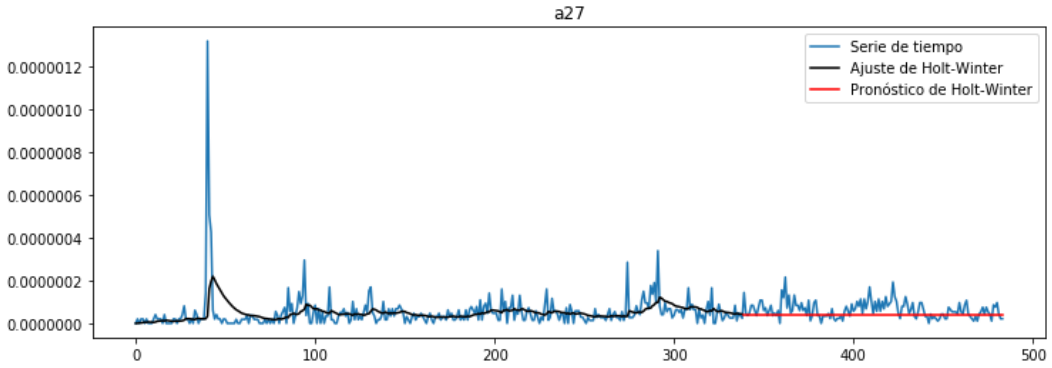

a03
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.000010
0.000008
0.000006
0.000004
0.000002
0.000000
0
100
200
300
400

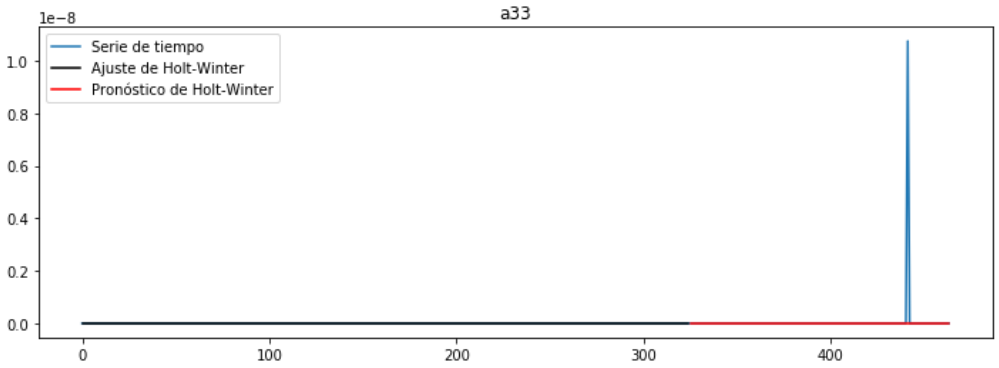

a05
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000175
0.0000150
0.0000125
0.0000100
0.0000075
0.0000050
0.0000025
0.0000000
0
100
200
300
400
500

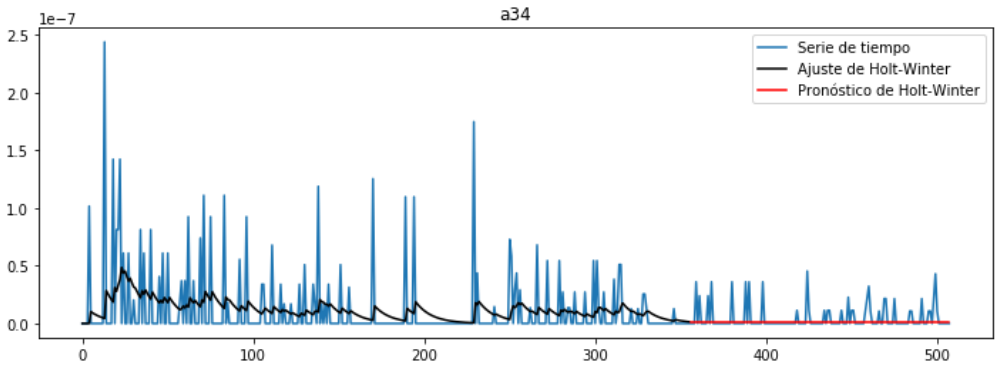

a15-a16
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.000006
0.000005
0.000004
0.000003
0.000002
0.000001
0.000000
0
100
200
300
400

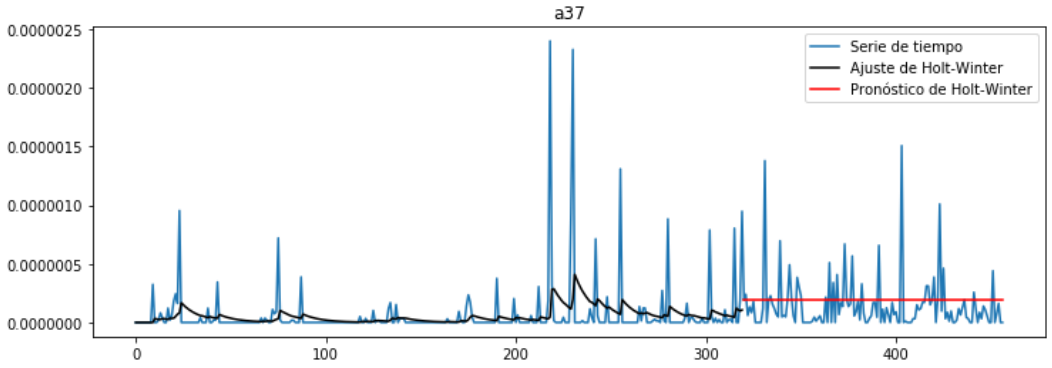

a17.0
1e−7
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400

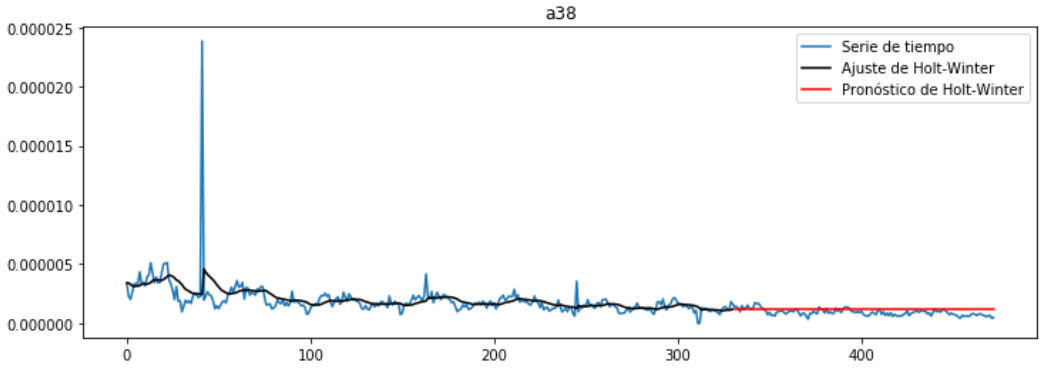

a23
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000014
0.0000012
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
0
100
200
300
400

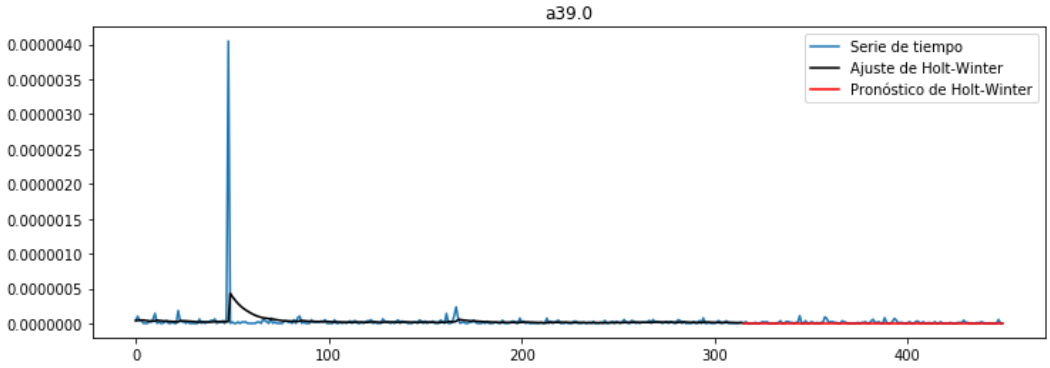

a27
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000012
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
0
100
200
300
400
500

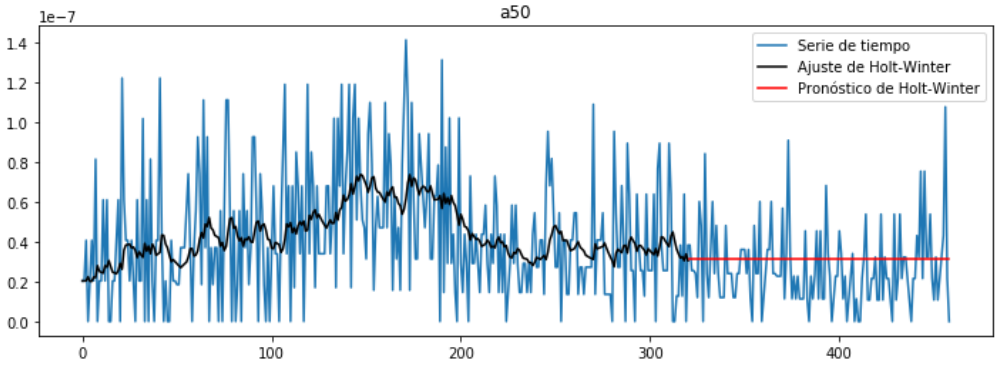

a33
1e−8
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400

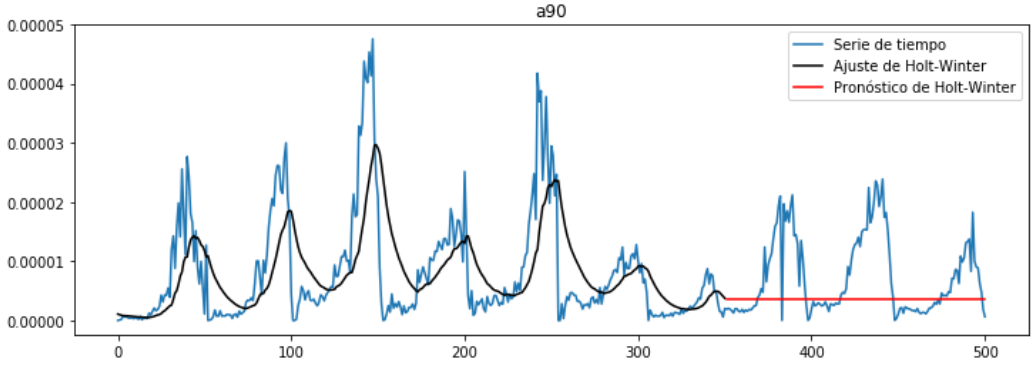

a34
1e−7
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
2.5
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400
500

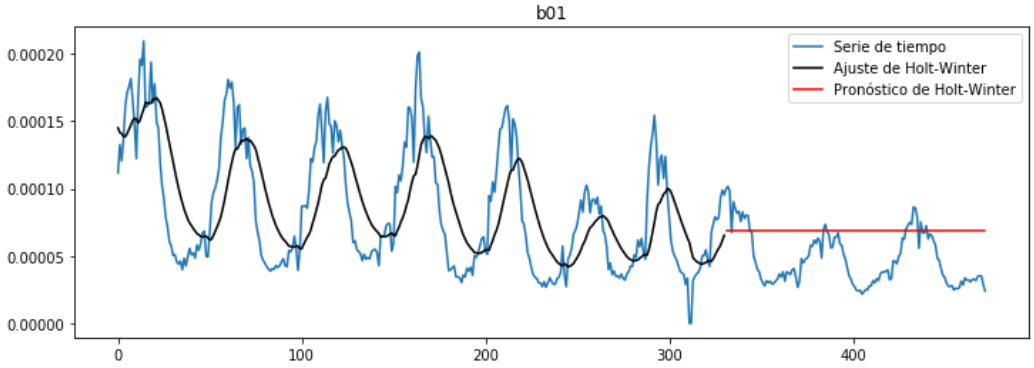

a37
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
0
100
200
300
400

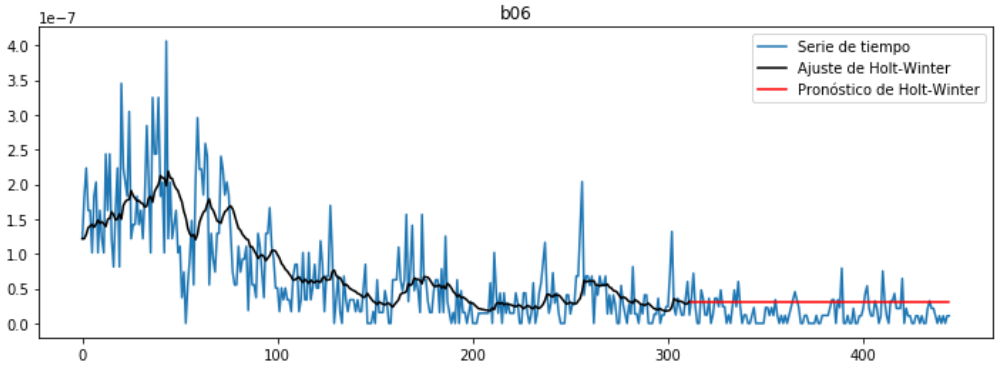

a38
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.000025
0.000020
0.000015
0.000010
0.000005
0.000000
0
100
200
300
400

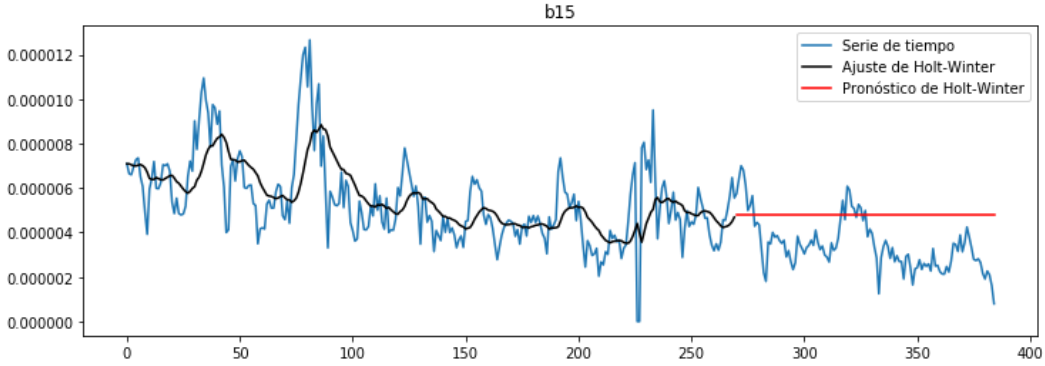

a39.0
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000040
0.0000035
0.0000030
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0.0000000
0
100
200
300
400

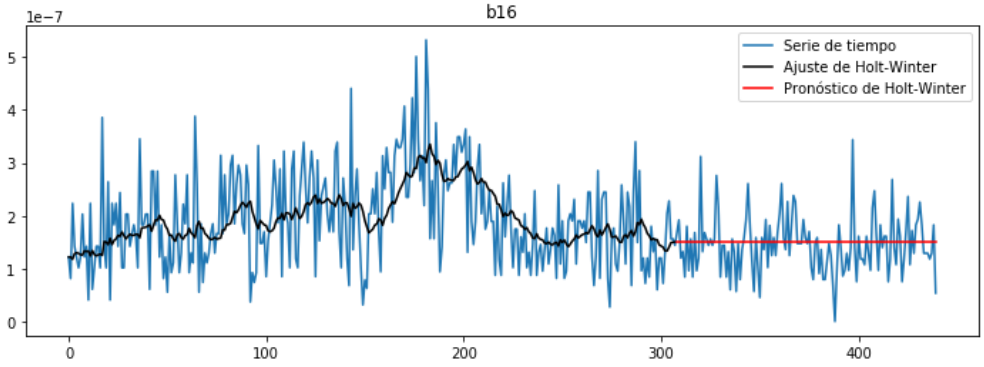

a50
1e−7
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400

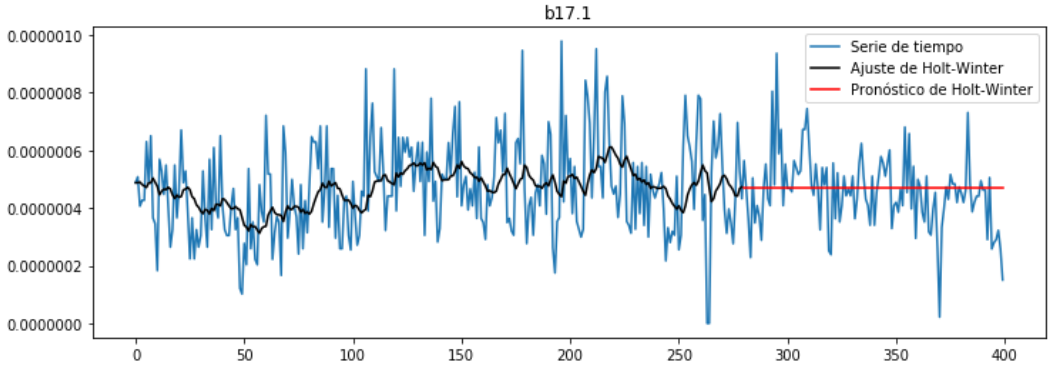

a90
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.00005
0.00004
0.00003
0.00002
0.00001
0.00000
0
100
200
300
400
500

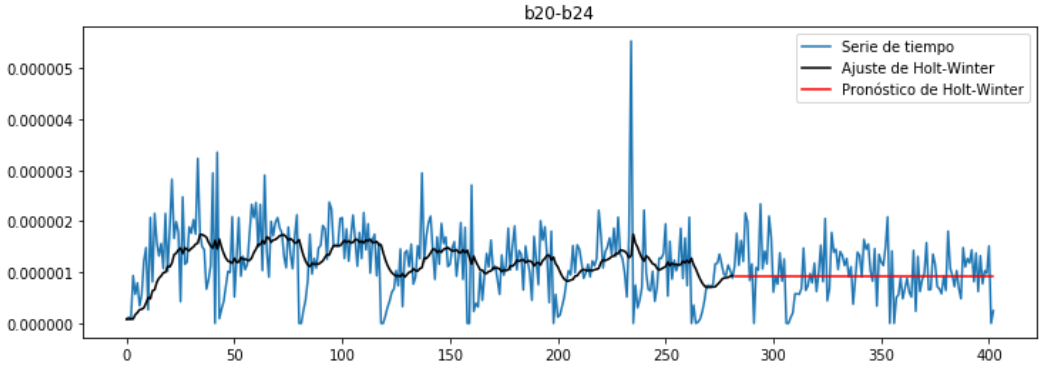

b01
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.00020
0.00015
0.00010
0.00005
0.00000
0
100
200
300
400

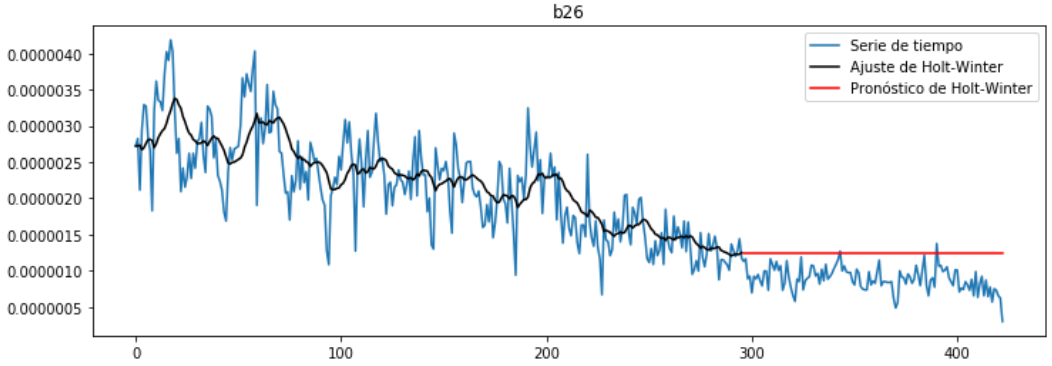

b06
1e−7
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400

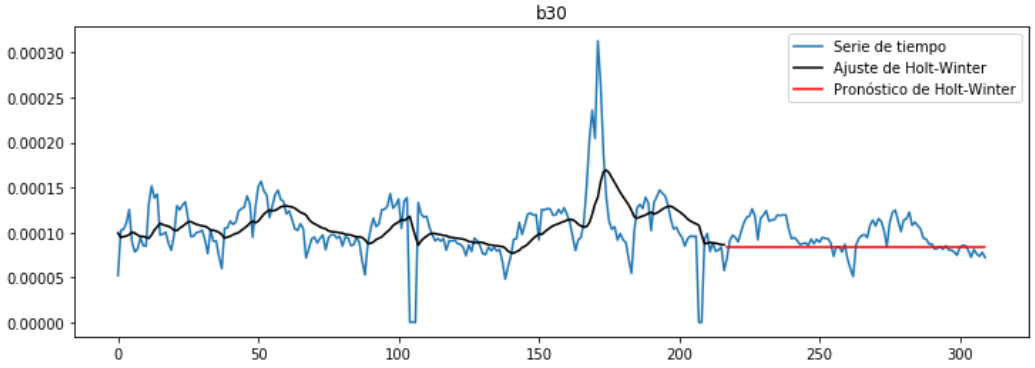

b15
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.000012
0.000010
0.000008
0.000006
0.000004
0.000002
0.000000
0
50
100
150
200
250
300
350
400

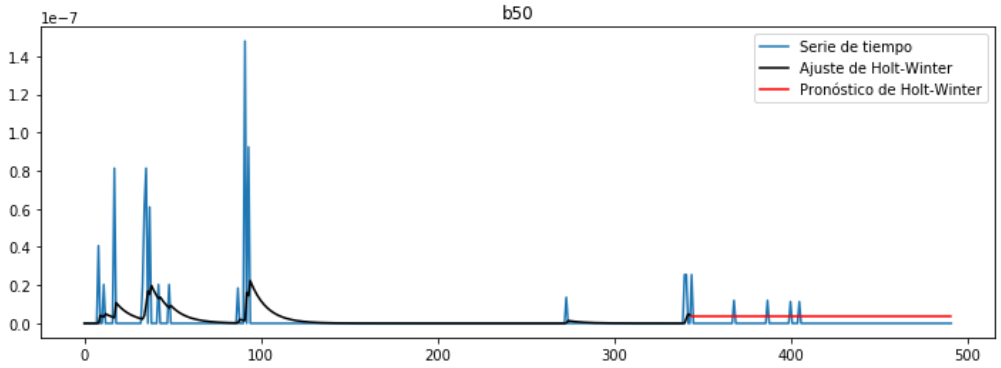

b16
1e−7
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
5
4
3
2
1
0
0
100
200
300
400

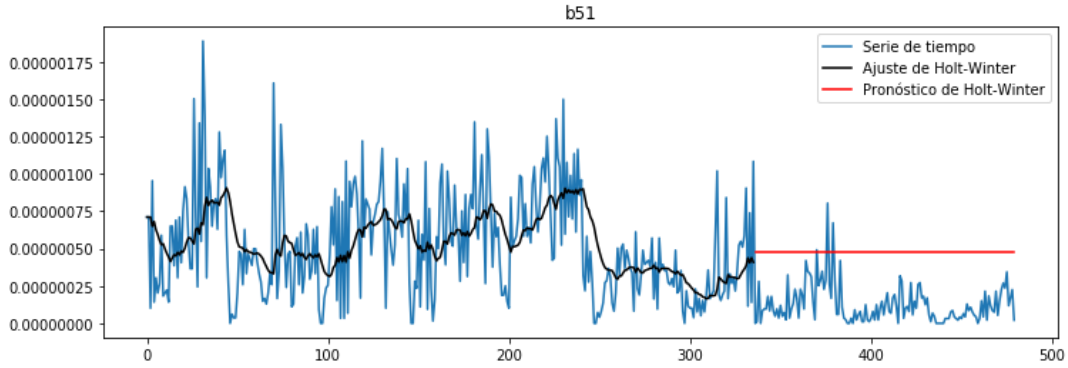

b17.1
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
0
50
100
150
200
250
300
350
400

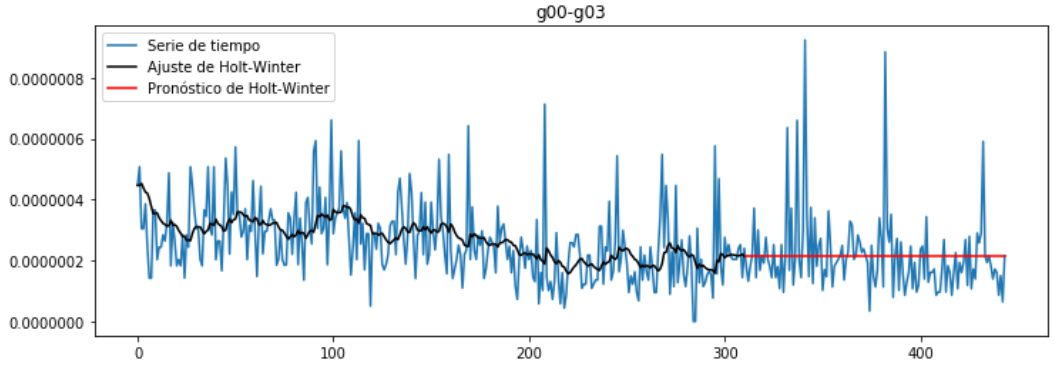

b20-b24
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.000005
0.000004
0.000003
0.000002
0.000001
0.000000
0
50
100
150
200
250
300
350
400

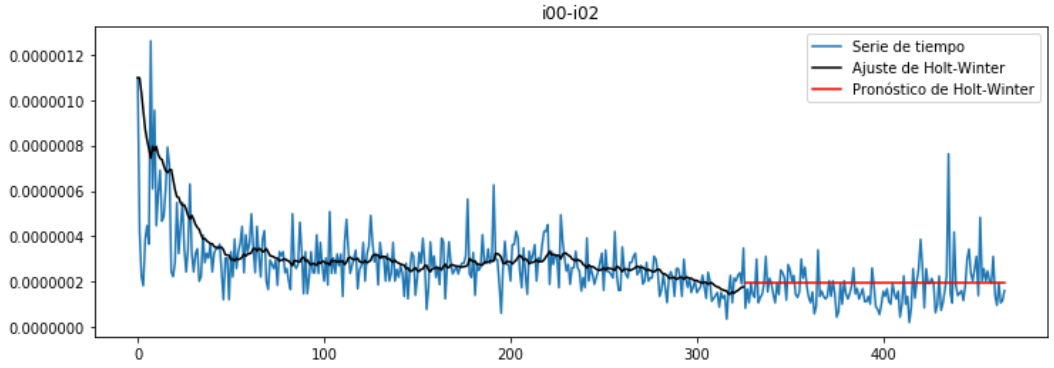

b26
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000040
0.0000035
0.0000030
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0
100
200
300
400

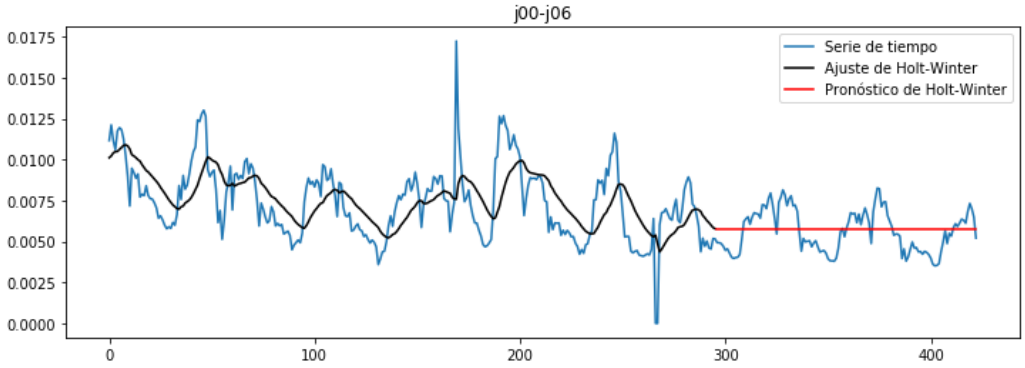

b30
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
0.00000
0
50
100
150
200
250
300

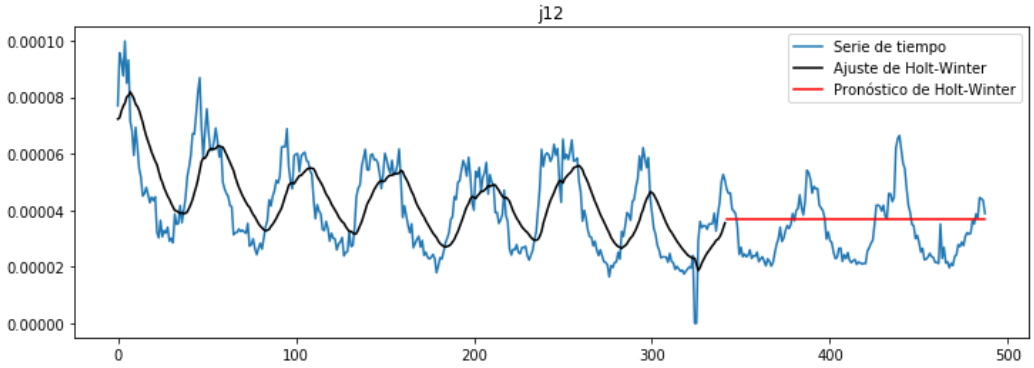

b50
1e−7
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter

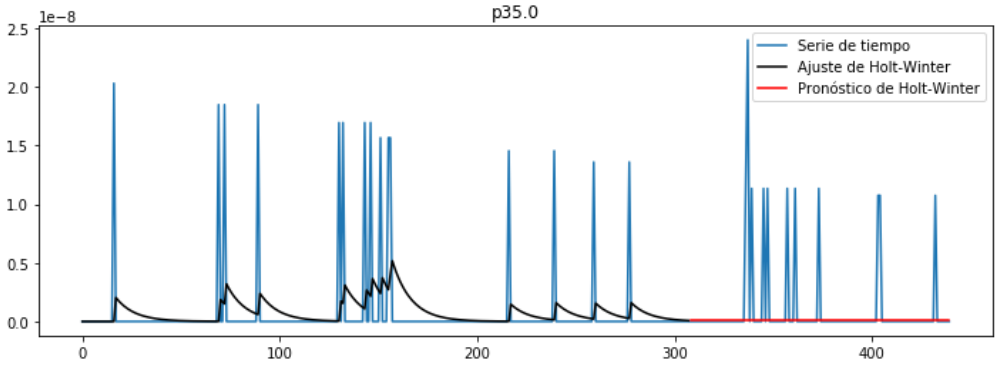

b51
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter

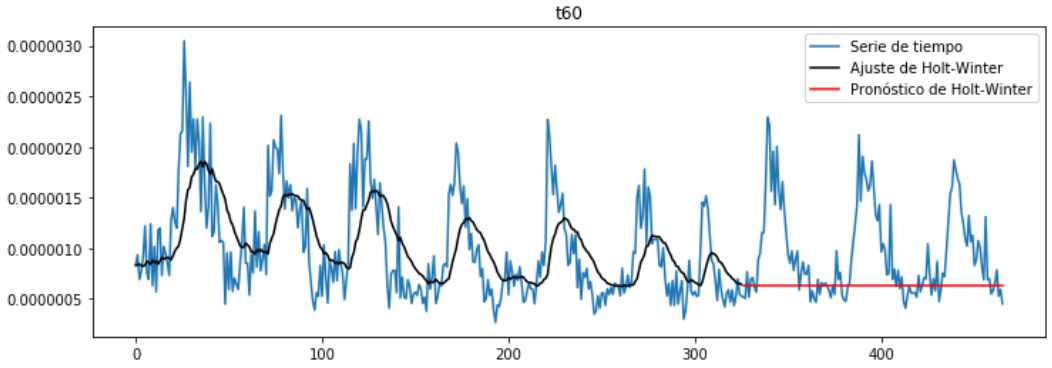

g00-g03
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter

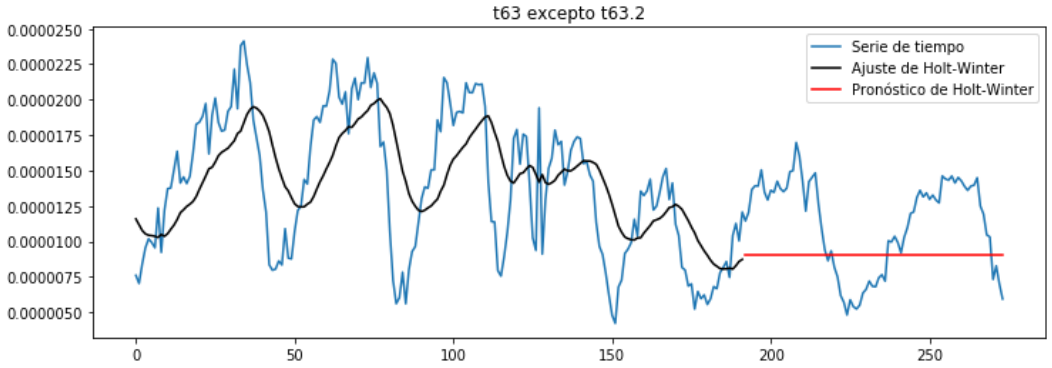

i00-i02
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter

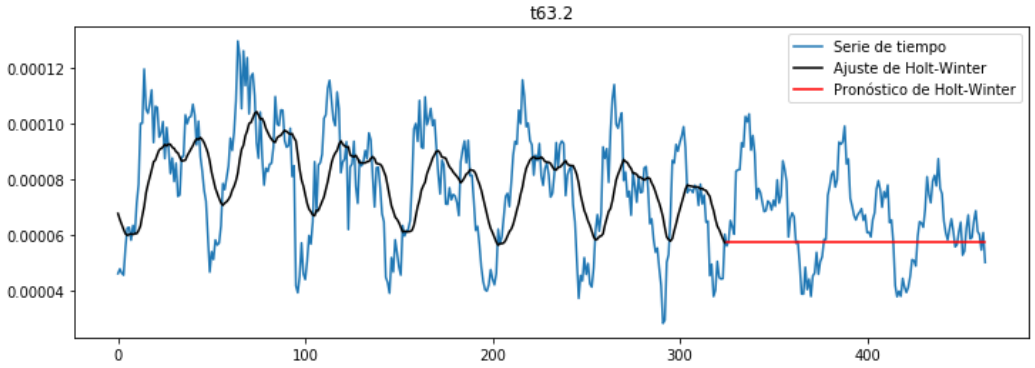

j00-j06
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0175
0.0150
0.0125
0.0100
0.0075
0.0050
0.0025
0.0000
0
100
200
300
400

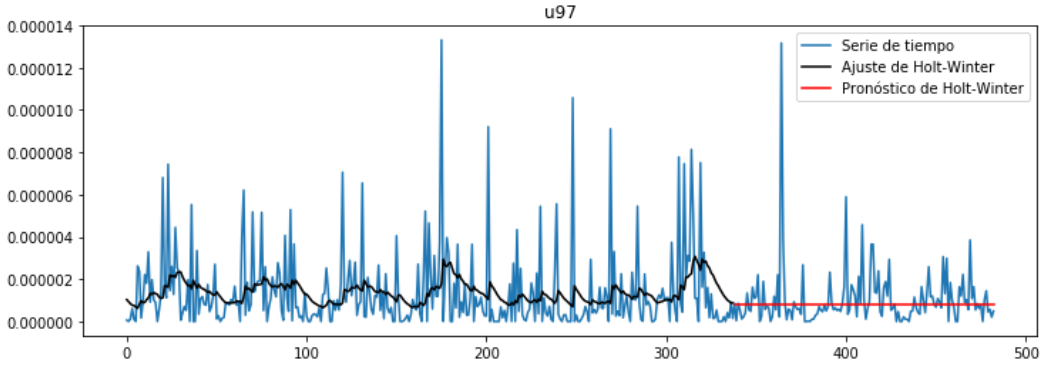

j12
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.00010
0.00008
0.00006
0.00004
0.00002
0.00000
0
100
200
300
400
500

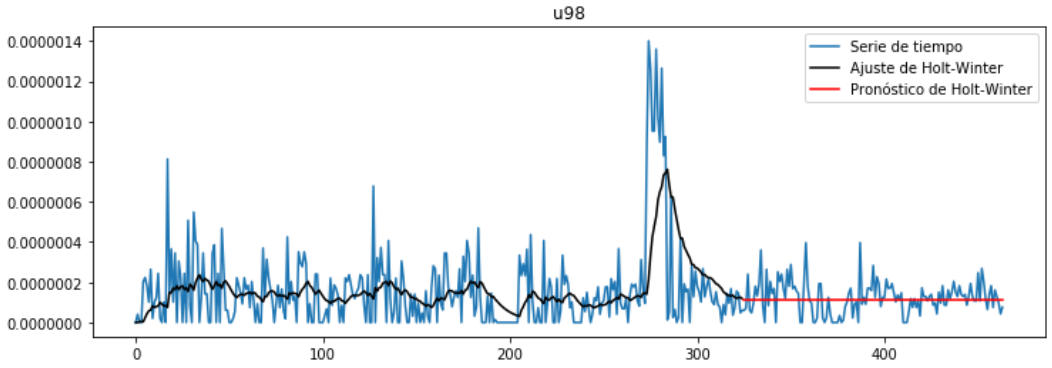

p35.0
1e−8
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
2.5
2.0
1.5
1.0
0.5
0.0
0
100
200
300
400

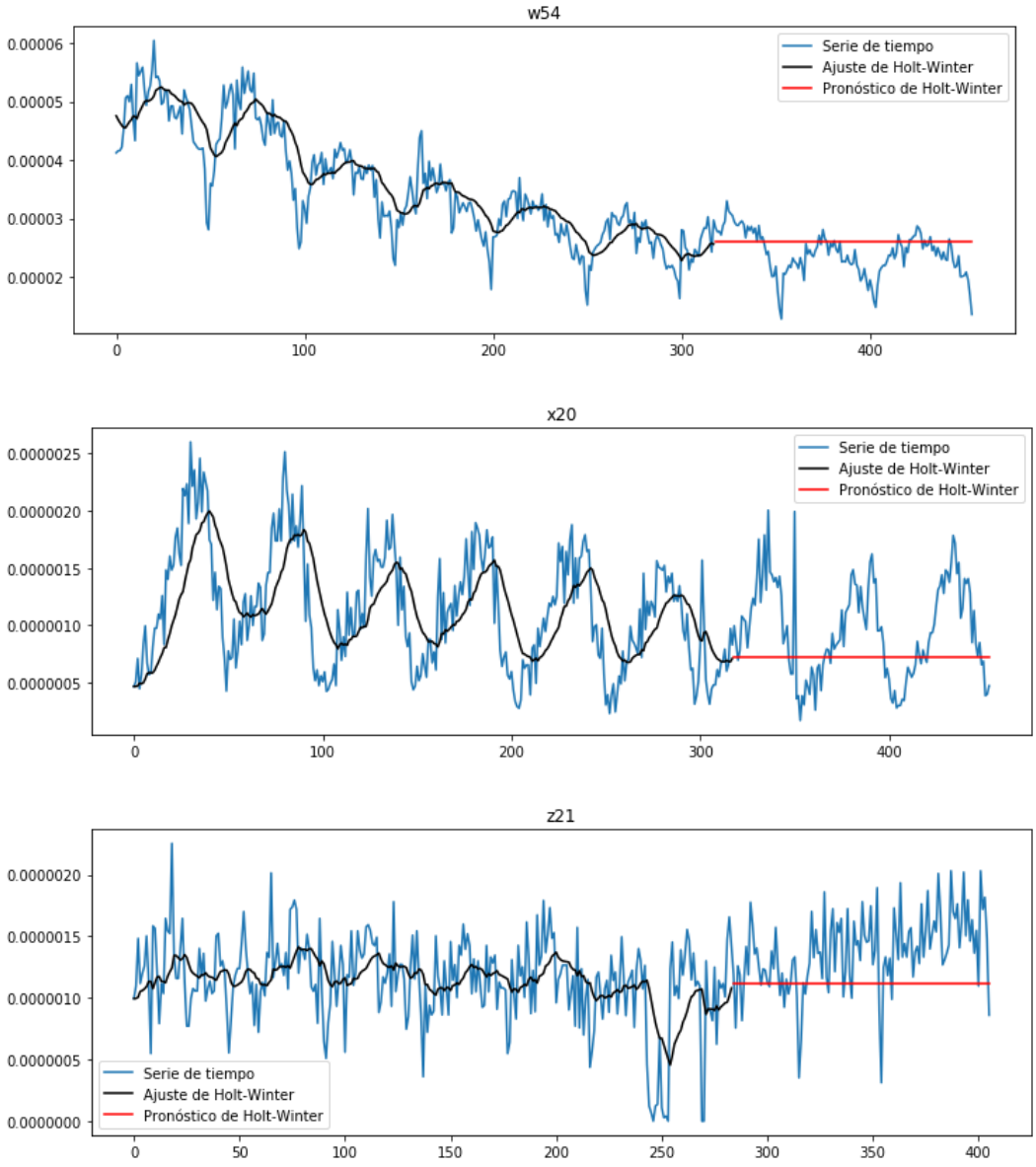

t60
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000030
0.0000025
0.0000020
0.0000015
0.0000010
0.0000005
0
100
200
300
400


t63 excepto t63.2
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000250
0.0000225
0.0000200
0.0000175
0.0000150
0.0000125
0.0000100
0.0000075
0.0000050
0
50
100
150
200
250


t63.2
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.00012
0.00010
0.00008
0.00006
0.00004
0
100
200
300
400


u97
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.000014
0.000012
0.000010
0.000008
0.000006
0.000004
0.000002
0.000000
0
100
200
300
400
500


u98
Serie de tiempo
Ajuste de Holt-Winter
Pronóstico de Holt-Winter
0.0000014
0.0000012
0.0000010
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
0
100
200
300
400

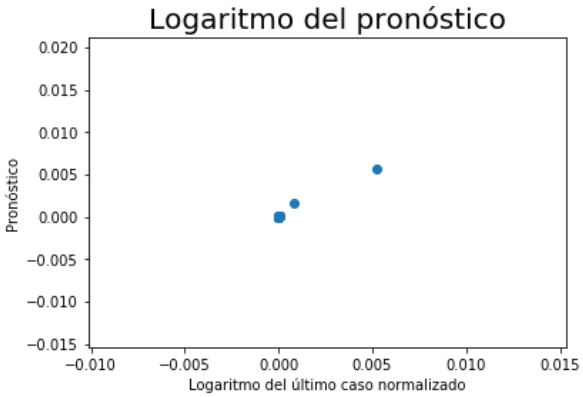


Que presenta muy buen ajuste respecto a los datos reales

```
a, b, r, p, e = stats.linregress(real, pronosticos)
print("y = f(x) = {:.4f} x + {:.4f}".format(a, b))
print("error", e)
print("p = ", p)
print("pendiente {:s}significativa".format("no " if p >= 0.05 else ""))
print("R^2", r**2)

plt.title('Logaritmo del pronóstico', fontsize = 20)
plt.xlabel('Logaritmo del último caso normalizado')
plt.ylabel('Pronóstico')
plt.scatter(real, pronosticos)
plt.show()
```

```
y = f(x) = 1.1122 x + 0.0000
```

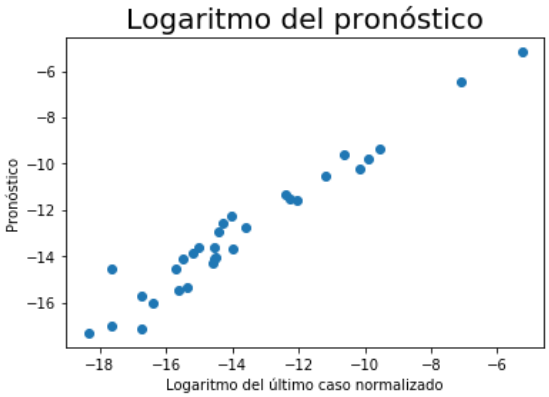
```
error 0.019116083562680818
p =  5.559074078899032e-38
pendiente significativa
R^2 0.9891871972161727
```



Lo que se evidencia claramente al utilizar escala logarítmica en los datos

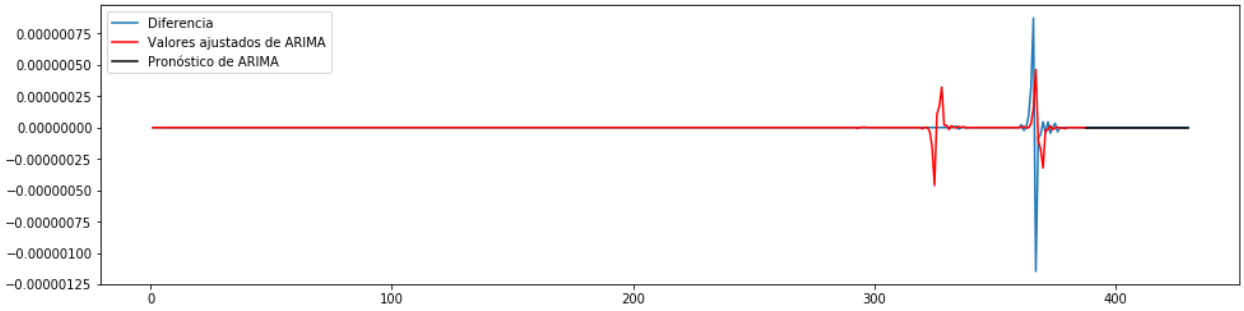
```
plt.title('Logaritmo del pronóstico', fontsize = 20)
plt.xlabel('Logaritmo del último caso normalizado')
plt.ylabel('Pronóstico')
plt.scatter(np.log(real), np.log(pronosticos))
plt.show()
```

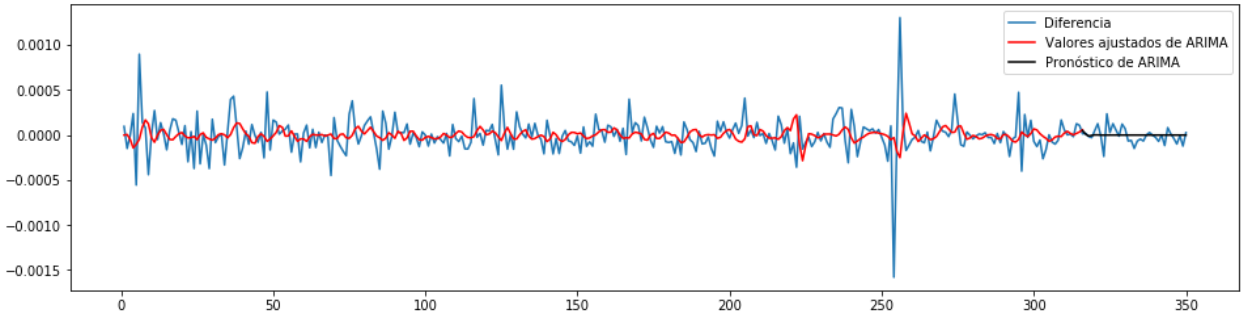
```
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\ipykernel_launcher.py:4:
RuntimeWarning: divide by zero encountered in log
  after removing the cwd from sys.path.
```

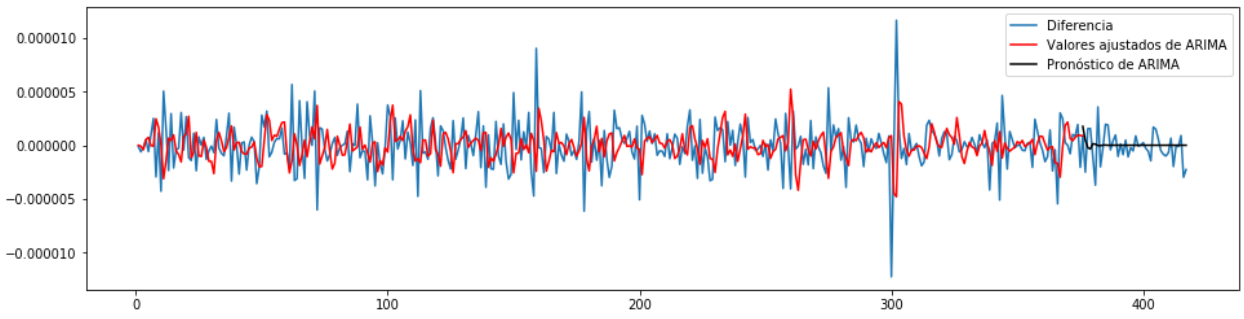


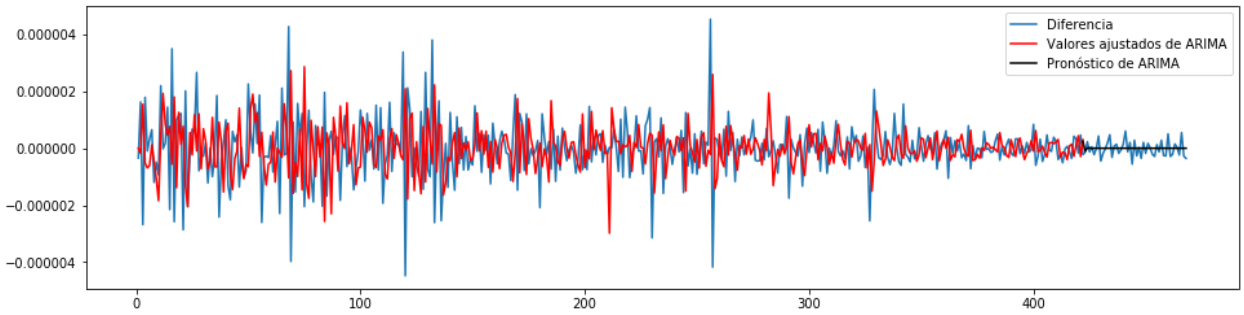

```
# https://machinelearningmastery.com/time-series-forecast-uncertainty-using-confidence-intervals-python
# https://machinelearningmastery.com/make-sample-forecasts-arima-python/
from statsmodels.tsa.arima_model import ARIMA

def difference(dataset, interval=1):
    diff = list()
    for i in range(interval, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
```

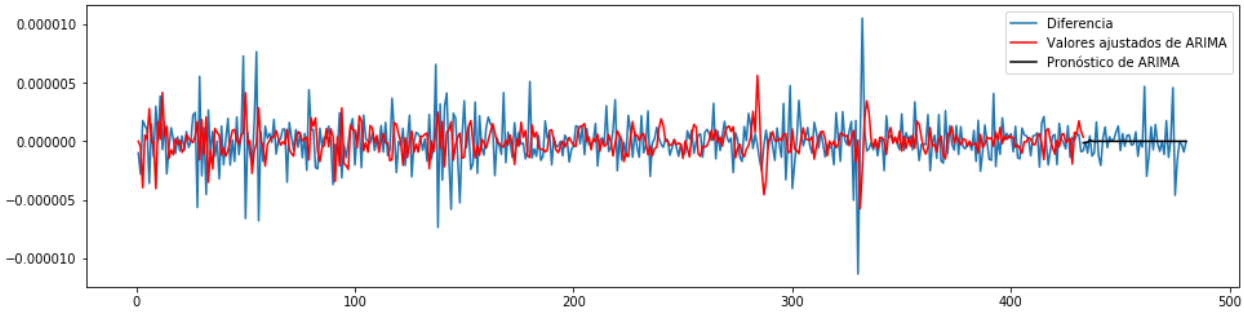
```
    return pd.Series(diff)

for i in range(len(ciesTS)):
    temp = ciesTS.iloc[i, : 508]
    temp.dropna(inplace=True)
    train = round(0.9 * len(temp))
    ts = difference(np.asarray(temp), interval=len(temp) - train - 1)
    m = ARIMA(ts, order=(3, 1, 0))
    f = m.fit(trend = 'nc', disp = 0)
    pred = f.predict(start=train + 1, end=len(temp))
    tempDiff = temp - temp.shift()
    plt.figure(figsize=(16, 4))
    plt.plot(tempDiff)
    plt.plot(f.fittedvalues, color='red')
    plt.plot(range(train, len(temp)), pred, c='black')
    plt.legend(['Diferencia', 'Valores ajustados de ARIMA', 'Pronóstico de ARIMA'])
    plt.show()
```

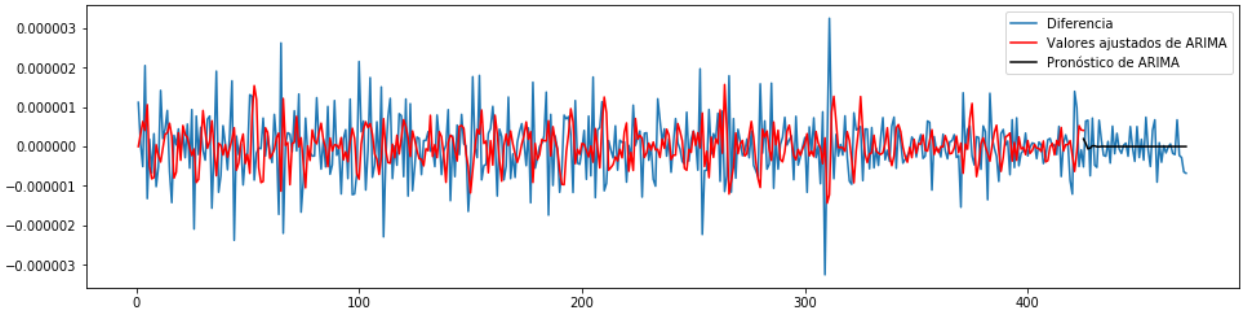


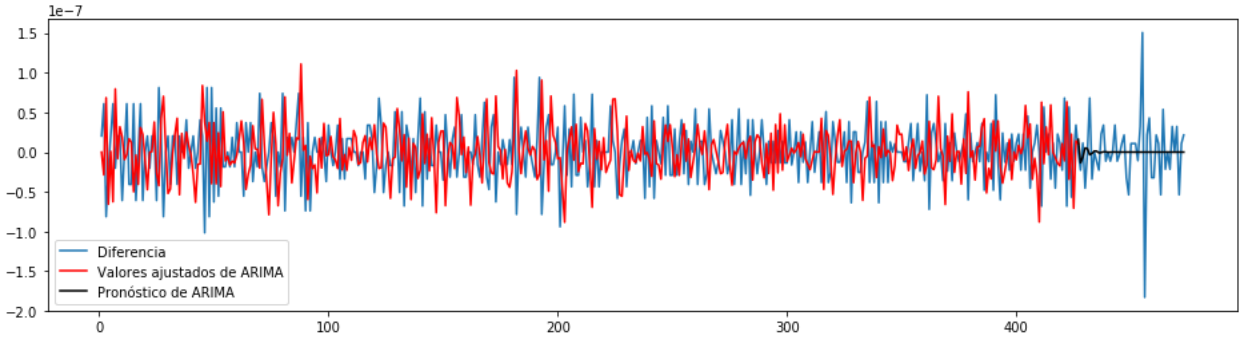


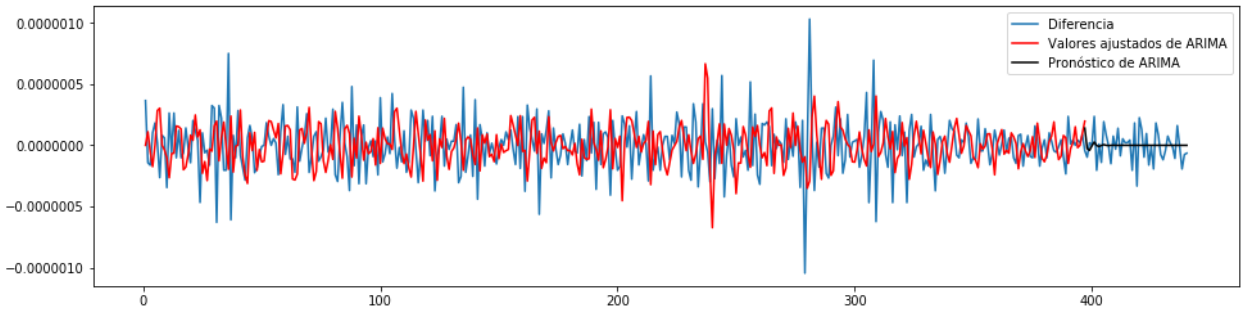


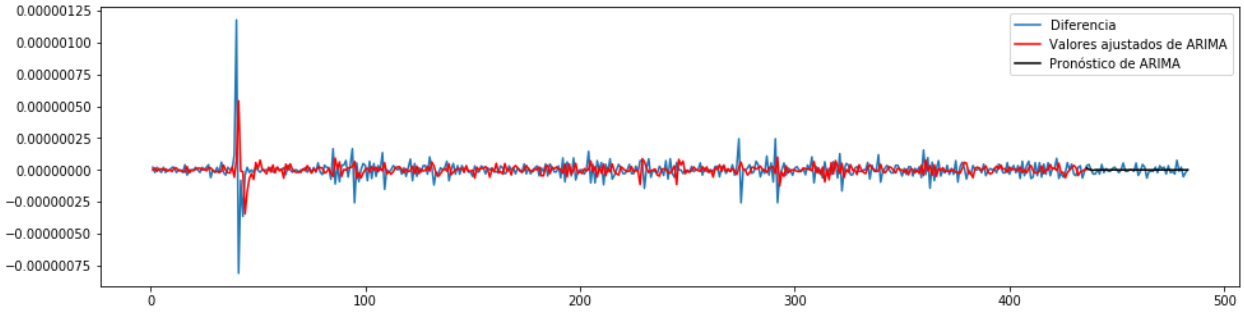

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000004
0.000002
0.000000
−0.000002
−0.000004
0
100
200
300
400

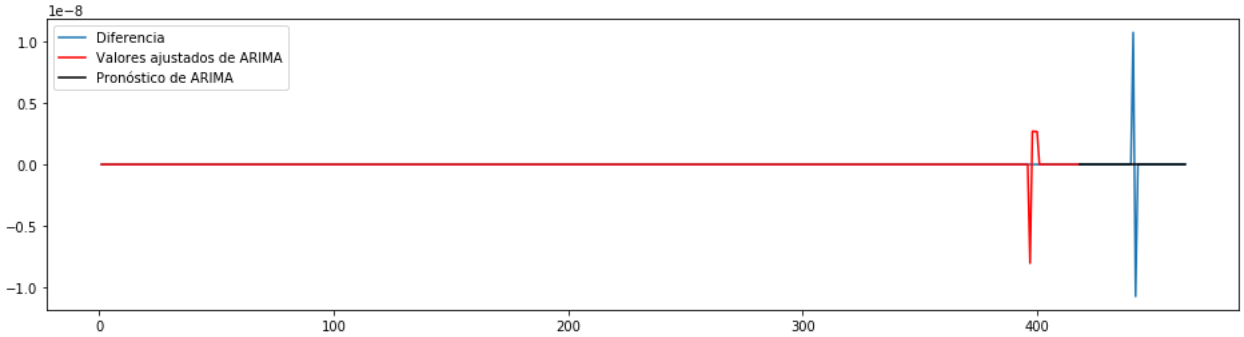

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000010
0.000005
0.000000
−0.000005
−0.000010
0
100
200
300
400
500

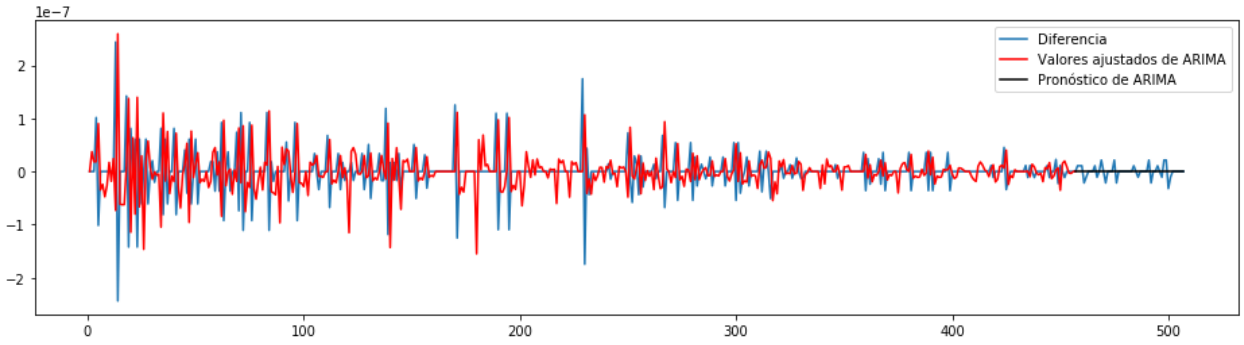

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000003
0.000002
0.000001
0.000000
−0.000001
−0.000002
−0.000003
0
100
200
300
400

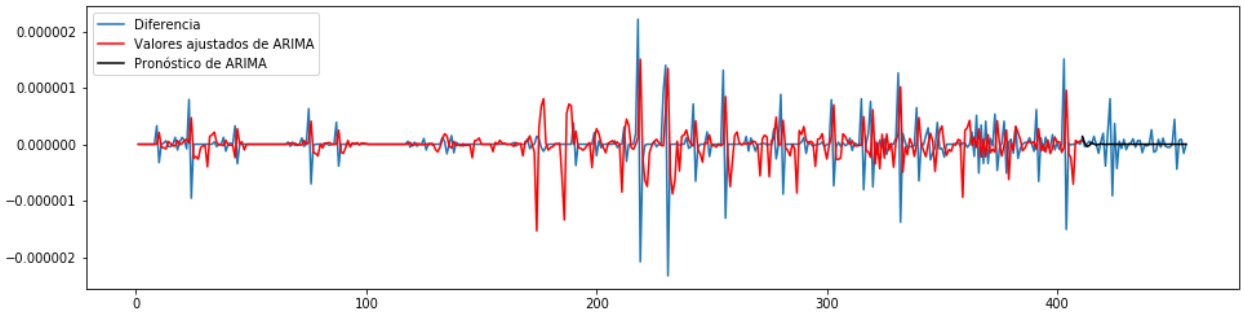

1e−7
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0
100
200
300
400

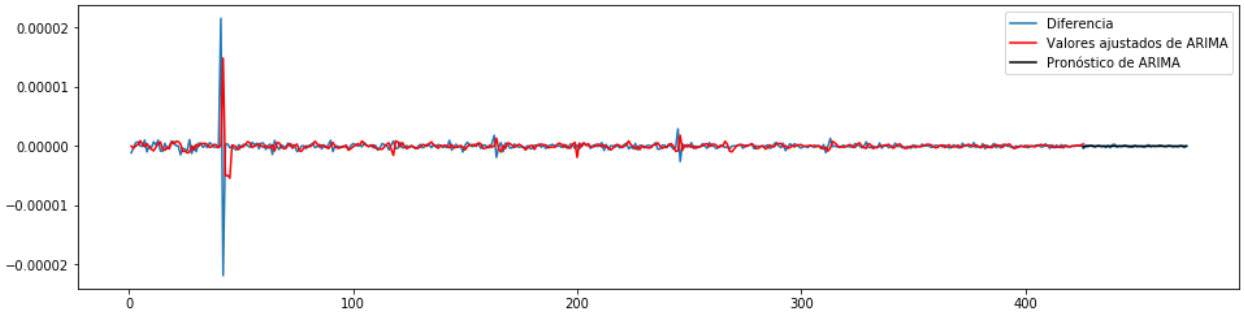

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

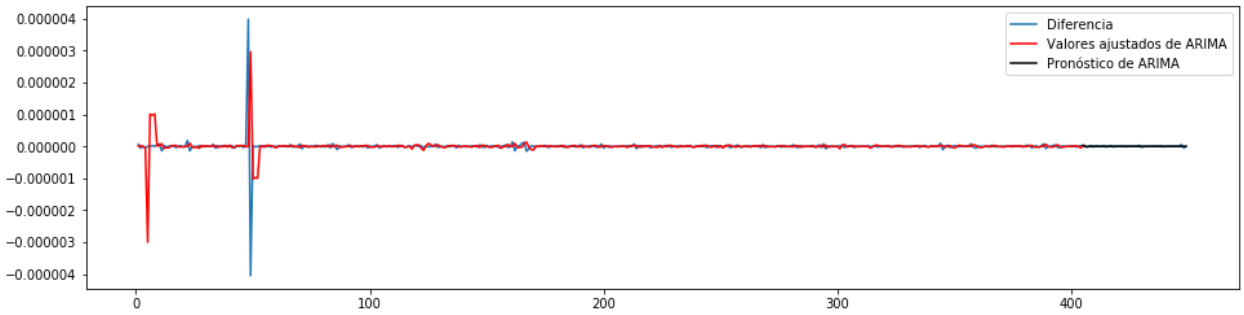

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

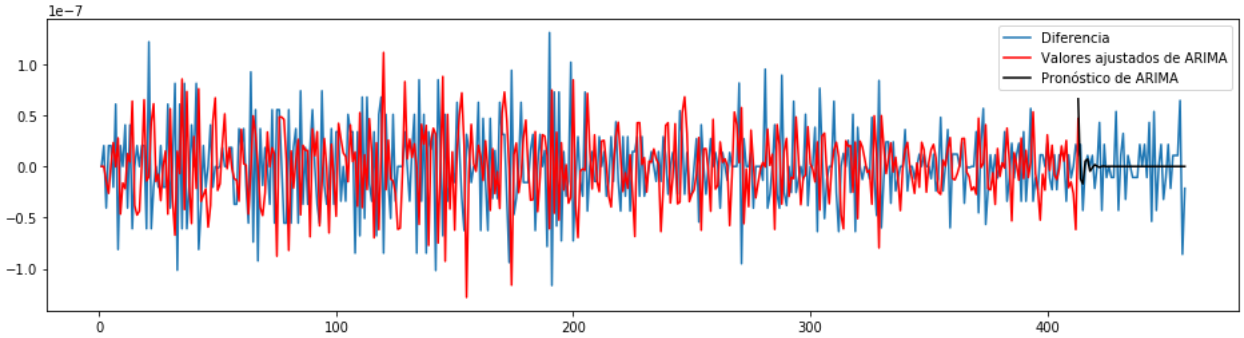

1e−8
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

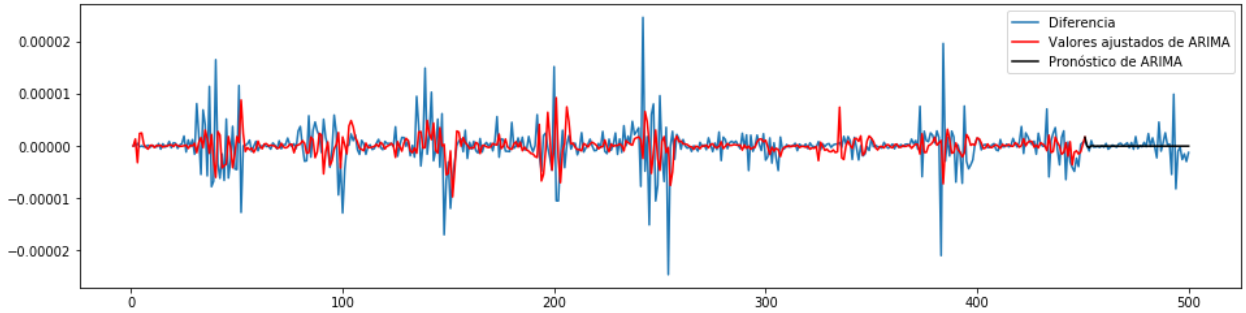

1e−7
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

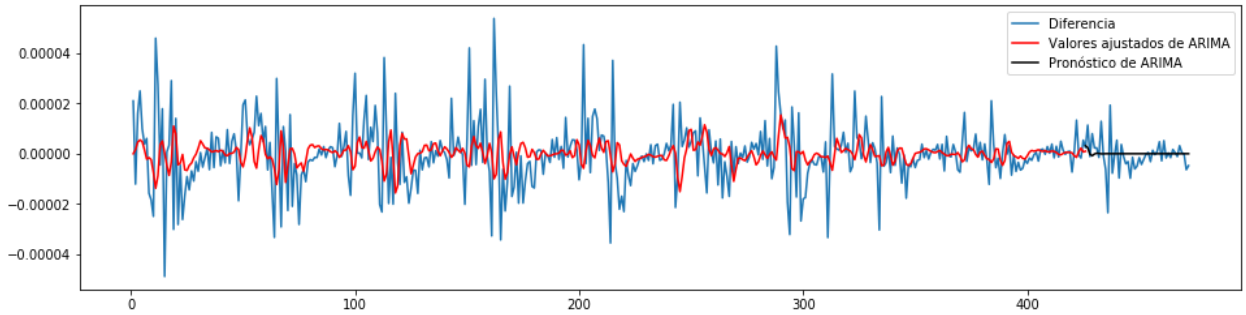

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000002
0.000001
0.000000
−0.000001
−0.000002
0
100
200
300
400

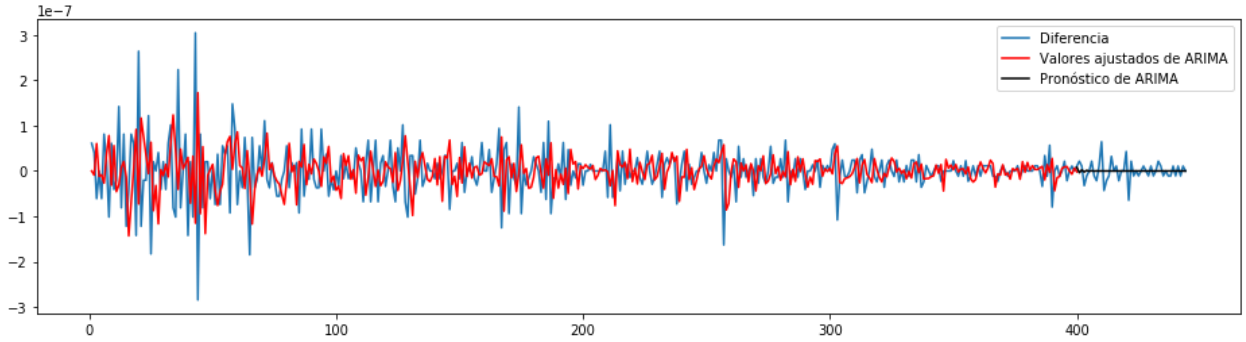

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.00002
0.00001
0.00000
−0.00001
−0.00002
0
100
200
300
400

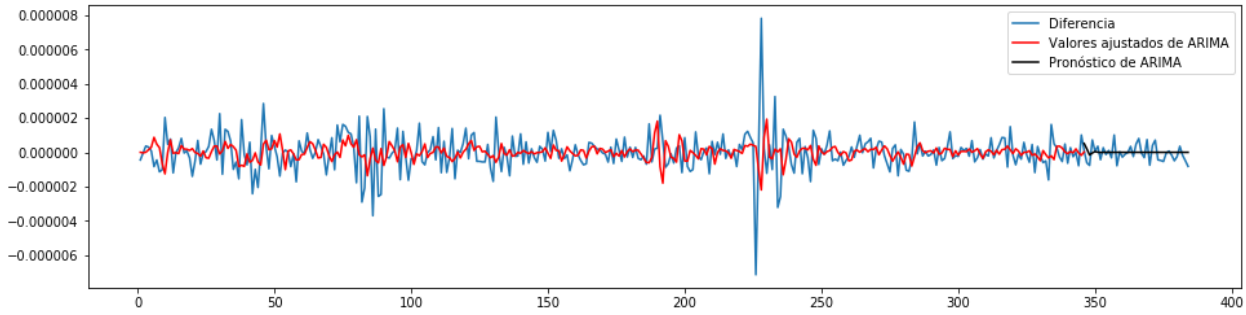

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000004
0.000003
0.000002
0.000001
0.000000
−0.000001
−0.000002
−0.000003
−0.000004
0
100
200
300
400

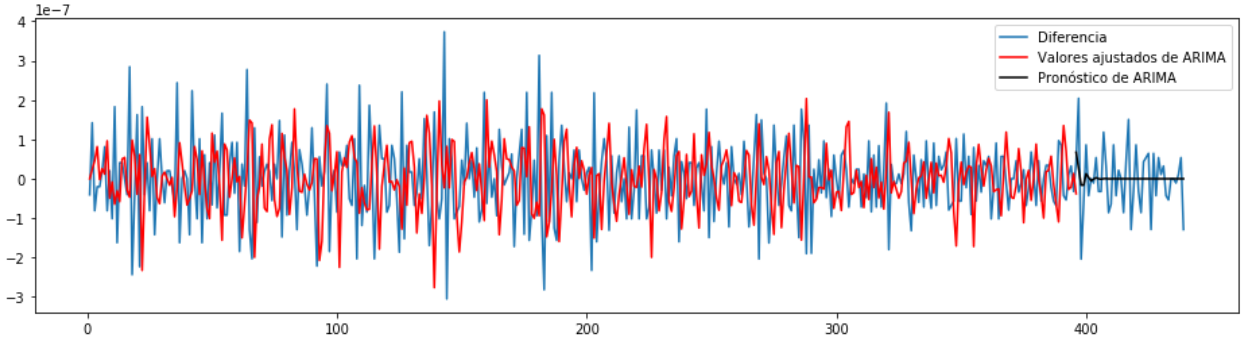

1e−7
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
1.0
0.5
0.0
−0.5
−1.0
0
100
200
300
400

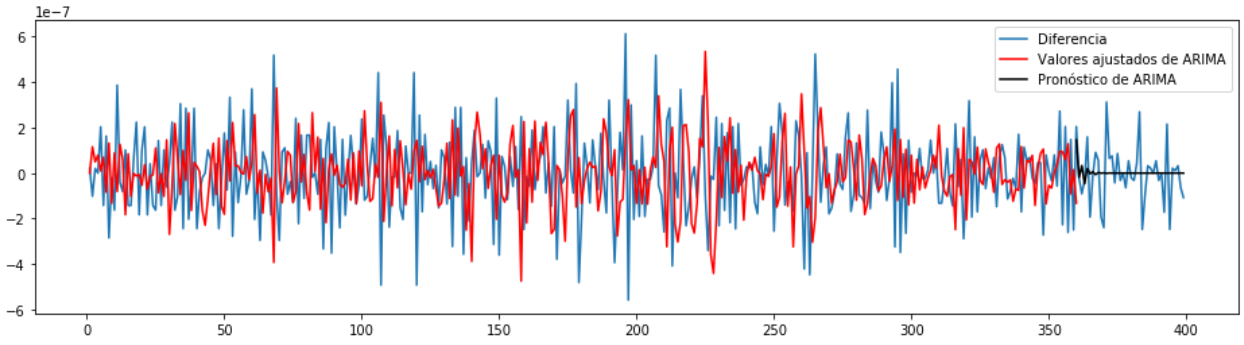

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.00002
0.00001
0.00000
−0.00001
−0.00002
0
100
200
300
400
500

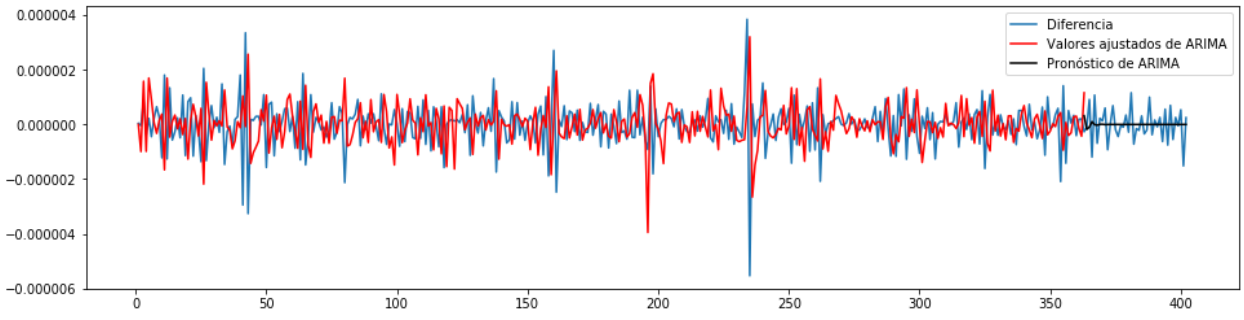

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.00004
0.00002
0.00000
−0.00002
−0.00004
0
100
200
300
400

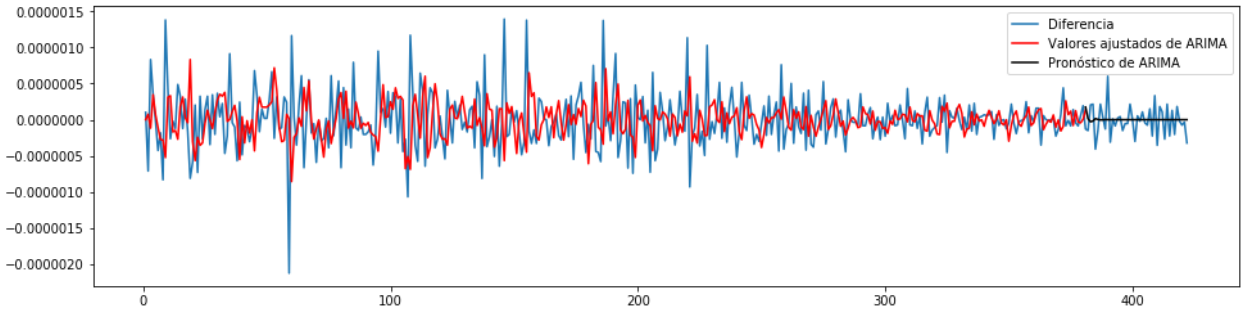

1e−7
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
3
2
1
0
−1
−2
−3
0
100
200
300
400

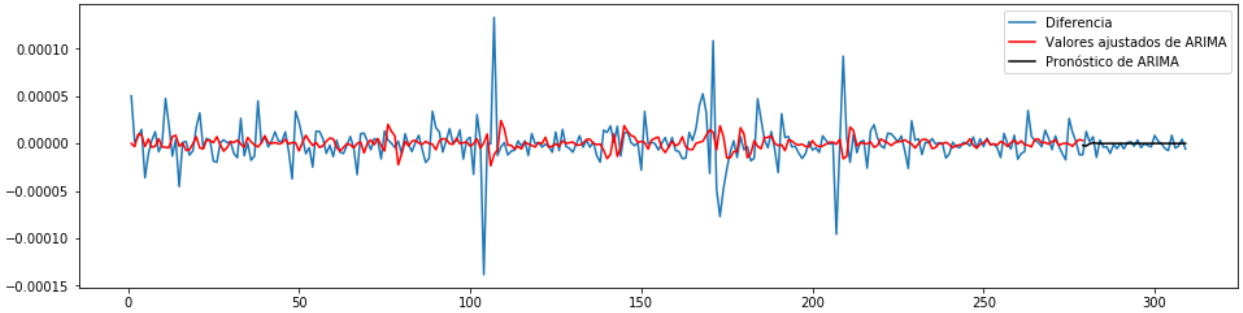

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000008
0.000006
0.000004
0.000002
0.000000
−0.000002
−0.000004
−0.000006
0
50
100
150
200
250
300
350
400

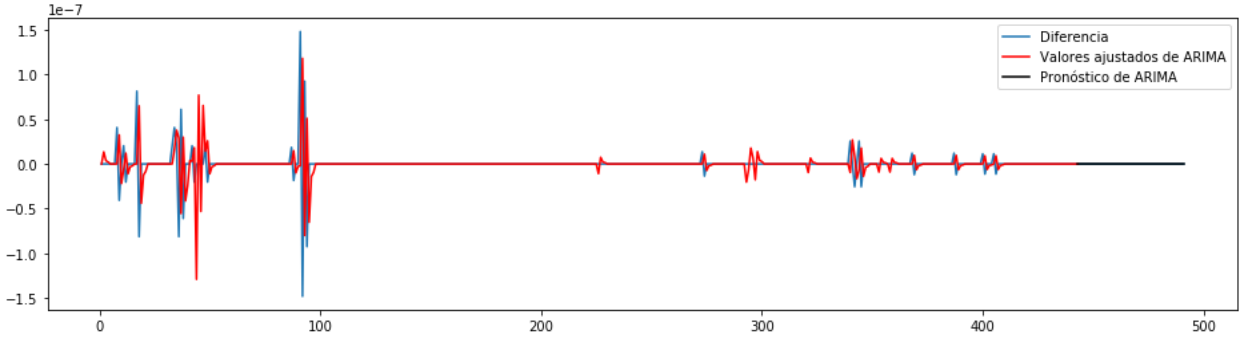

1e−7
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

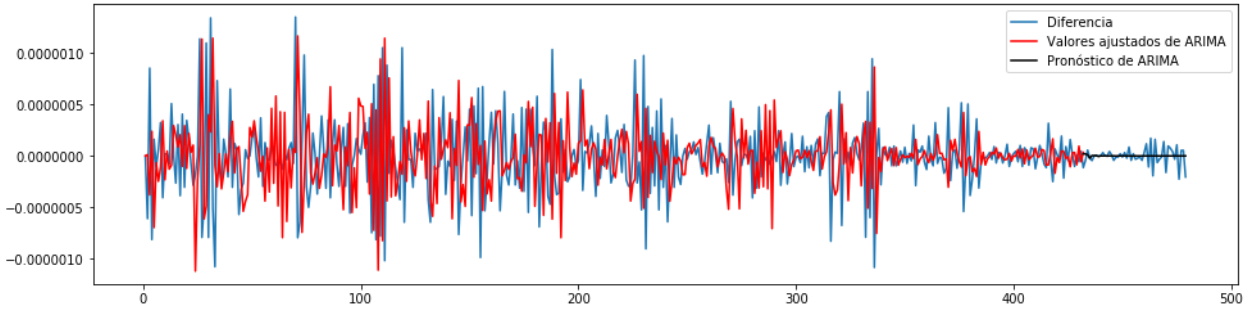

1e−7
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

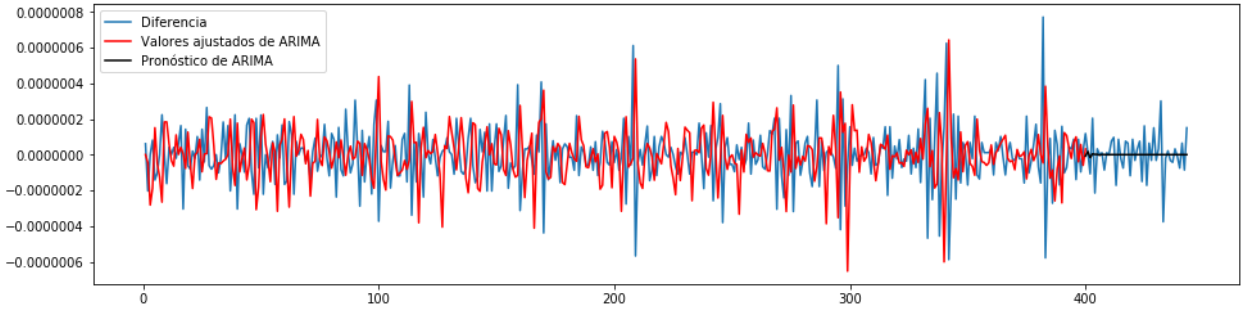

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

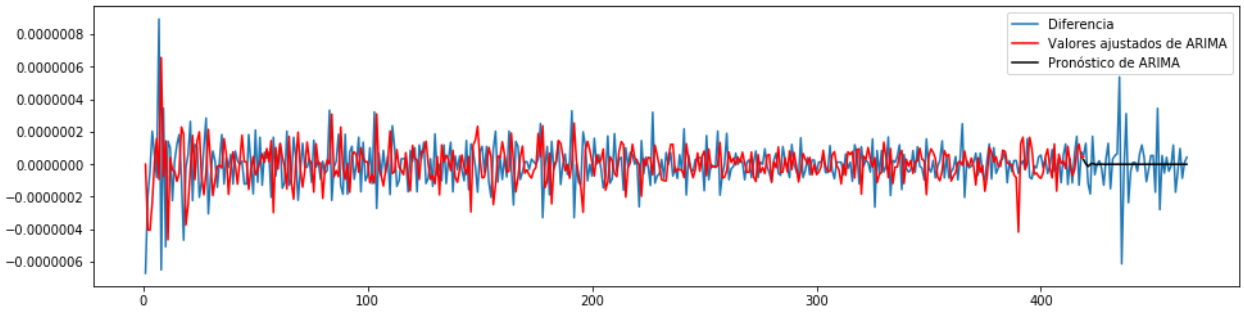

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

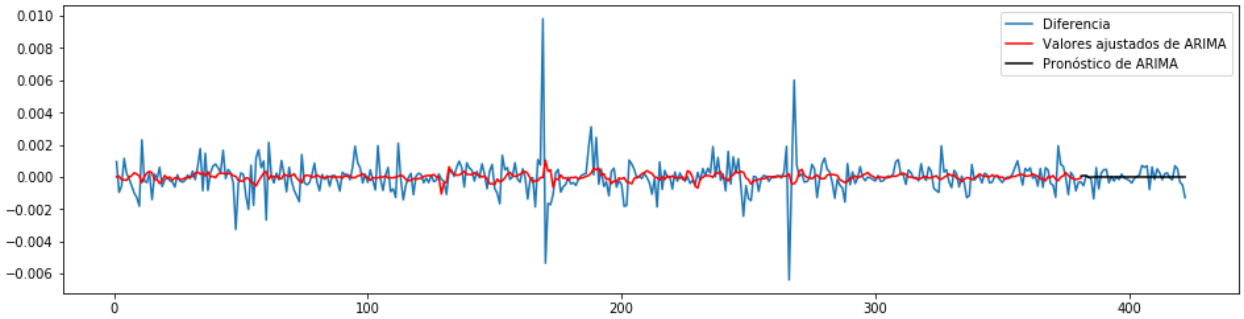

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

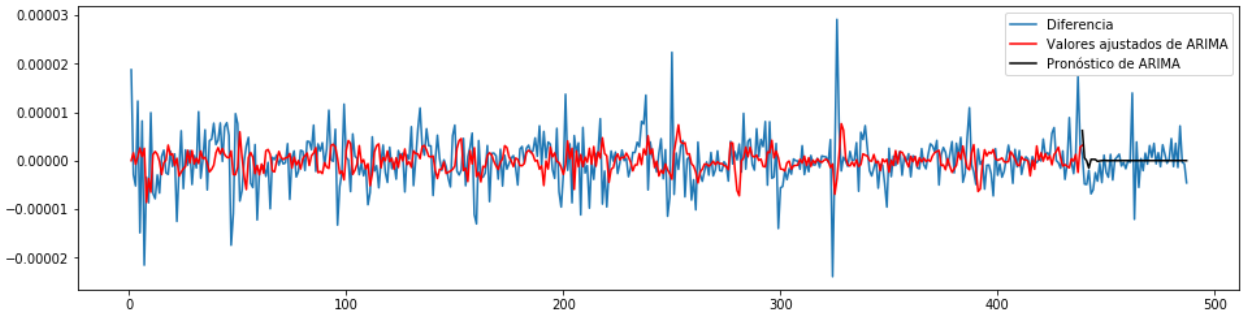

1e−7
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

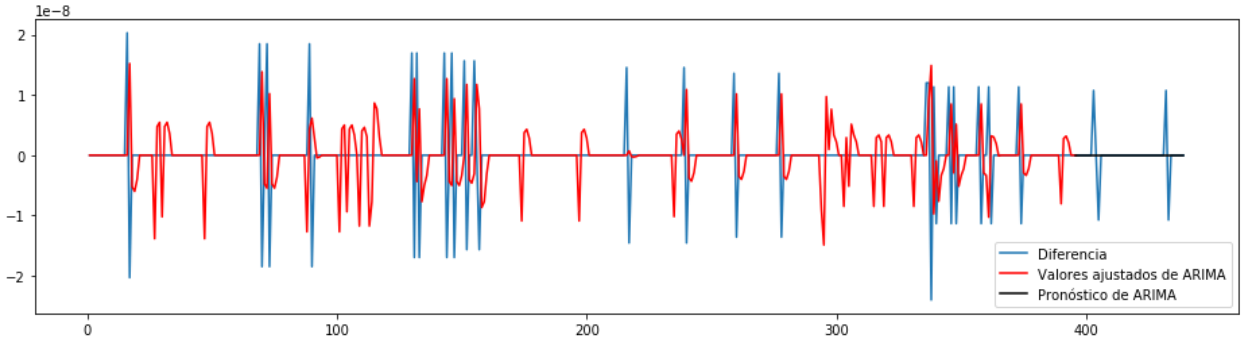

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

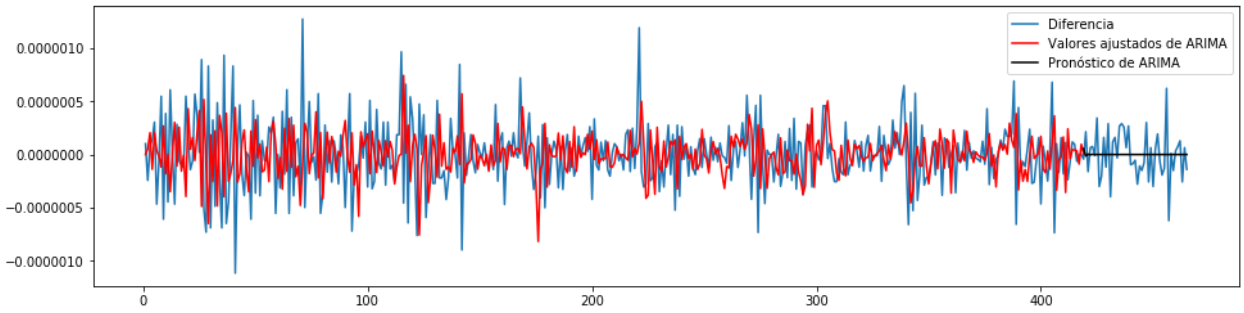

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA

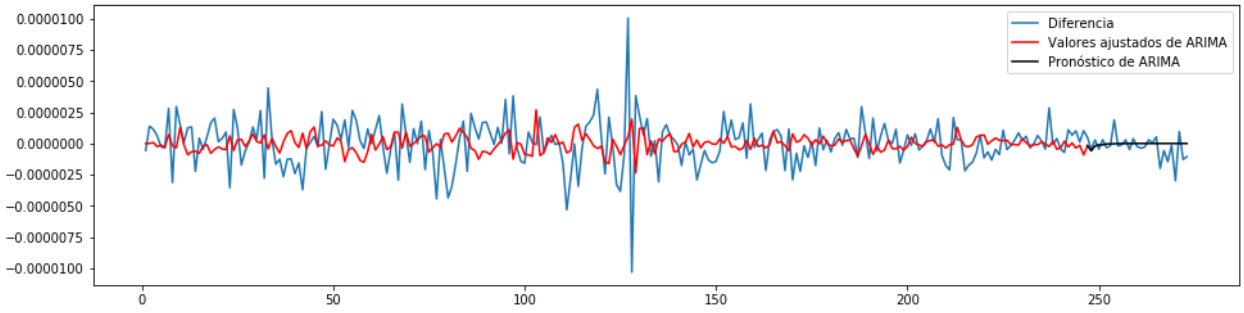

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.0000008
0.0000006
0.0000004
0.0000002
0.0000000
−0.0000002
−0.0000004
−0.0000006
0
100
200
300
400

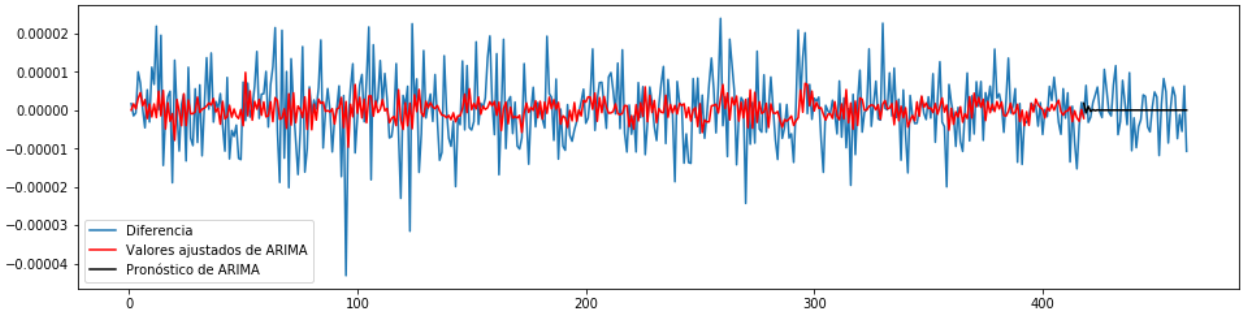

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.010
0.008
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
0
100
200
300
400

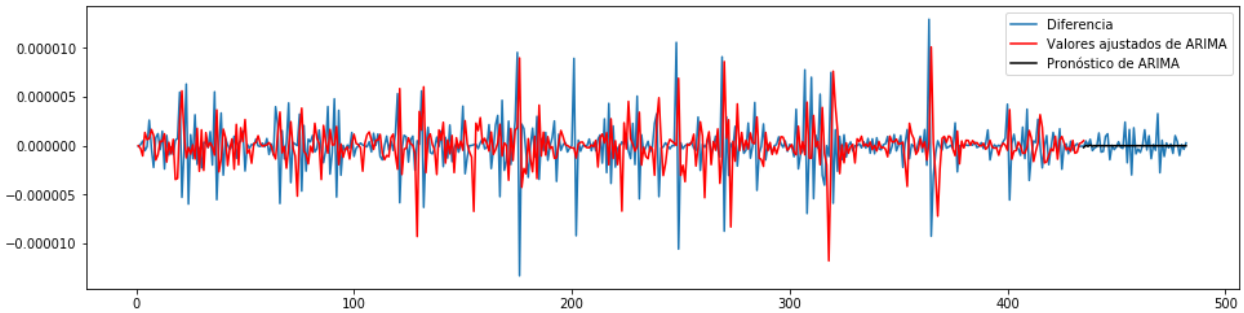

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.00003
0.00002
0.00001
0.00000
−0.00001
−0.00002
0
100
200
300
400
500

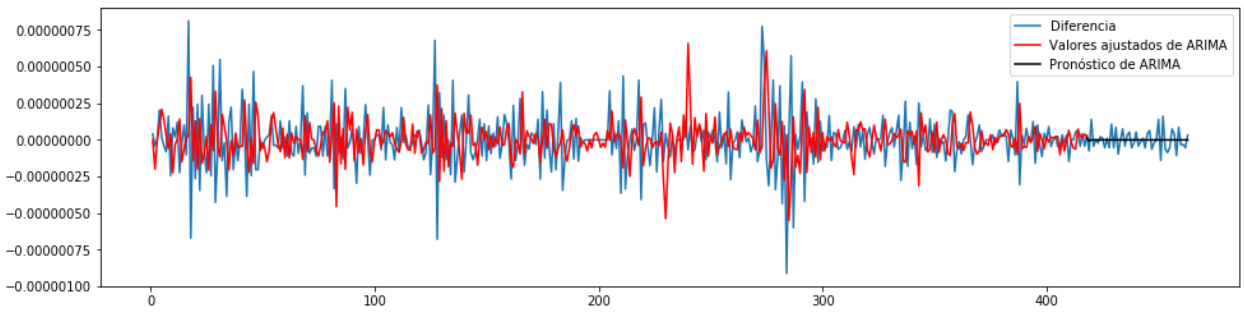

1e−8
2
1
0
−1
−2
Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0
100
200
300
400

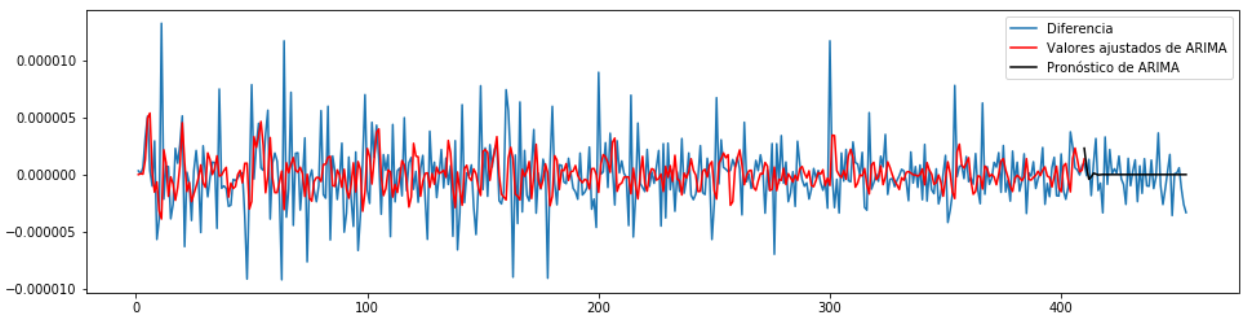

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.0000010
0.0000005
0.0000000
−0.0000005
−0.0000010
0
100
200
300
400

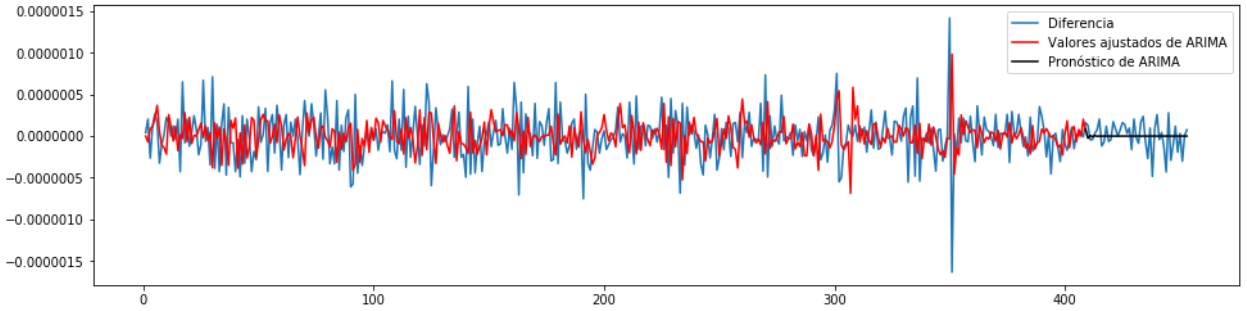

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.0000100
0.0000075
0.0000050
0.0000025
0.0000000
−0.0000025
−0.0000050
−0.0000075
−0.0000100
0
50
100
150
200
250

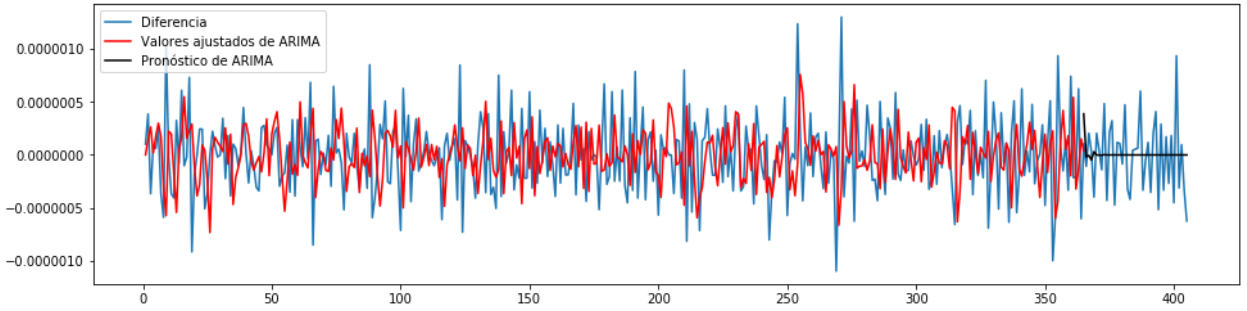

Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.00002
0.00001
0.00000
−0.00001
−0.00002
−0.00003
−0.00004
0
100
200
300
400


Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000010
0.000005
0.000000
−0.000005
−0.000010
0
100
200
300
400
500


Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.00000075
0.00000050
0.00000025
0.00000000
−0.00000025
−0.00000050
−0.00000075
−0.00000100
0
100
200
300
400


Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.000010
0.000005
0.000000
−0.000005
−0.000010
0
100
200
300
400


Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.0000015
0.0000010
0.0000005
0.0000000
−0.0000005
−0.0000010
−0.0000015
0
100
200
300
400


Diferencia
Valores ajustados de ARIMA
Pronóstico de ARIMA
0.0000010
0.0000005
0.0000000
−0.0000005
−0.0000010
0
50
100
150
200
250
300
350
400

Ciencia de datos

Práctica 10. Clasificación de datos con sklearn

Alberto Benavides

Primero se recuperan las características para las CIEs que contienen la pendiente, ordenada en el origen y las 52 autocorrelaciones correspondientes a retrasos de 1 a 52 semanas por serie de tiempo. Estos valores provienen de prácticas anteriores.

In [120]:
```
import pandas as pd

df = pd.read_csv("C:/Users/bena8/Documents/FIME/Tesis/Resultados/features.csv")
df.sample(3)
```

Out[120]:

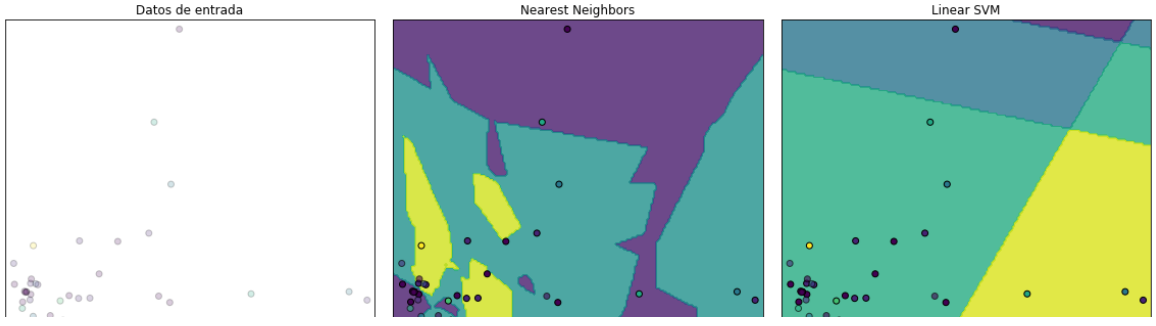
	m	b	ac1	ac2	ac3	ac4	ac5	ac6	ac7	ac8	...	ac44	
26	-2.654424e-10	3.204952e-07	0.110847	0.069672	0.107836	0.080559	0.128797	-0.027006	0.042914	-0.009807	...	0.029668	0.0
8	3.893878e-11	4.432440e-08	0.393243	0.260146	0.044702	0.032172	0.012605	0.025166	0.014480	0.009281	...	-0.017366	0.0
33	-5.001284e-08	8.796233e-05	0.881185	0.793629	0.694008	0.596382	0.456681	0.319222	0.189304	0.062305	...	0.314212	0.3

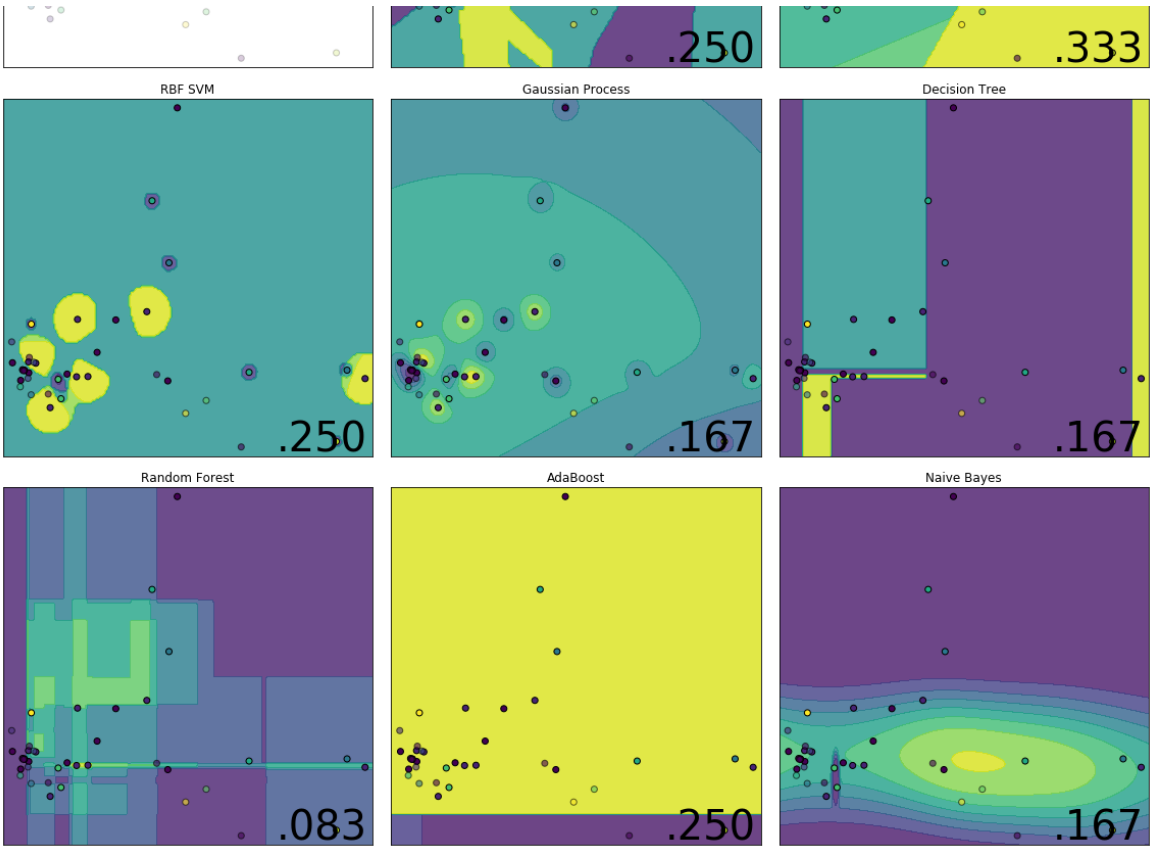
3 rows × 55 columns

Y de estas características, se eligen las 2 principales provenientes del análisis de componentes principales para hacer las clasificaciones

In [114]:
```
from math import ceil, sqrt
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from numpy import isnan, nan, arange, meshgrid, c_, unique
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
import matplotlib.cm as cm

x = df.iloc[:, :-1]
y = df.iloc[:, -1]
y_copy = y
x_copy = x

y = y.astype(str).str[0]
```

```
classnames, y = unique(y, return_inverse=True)

x = StandardScaler().fit_transform(x)
pca = PCA(n_components = 2) # pedimos uno bidimensional
X = pca.fit_transform(x)
# código de https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html
names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
    SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
    DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max_features=1), \
    AdaBoostClassifier(), GaussianNB()]
k = int(ceil(sqrt(len(classifiers) + 1)))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42) # división
x_min, x_max = X[:, 0].min() - .5, X[:, 0].max() + .5
y_min, y_max = X[:, 1].min() - .5, X[:, 1].max() + .5
xx, yy = meshgrid(arange(x_min, x_max, 0.1), arange(y_min, y_max, 0.1))
plt.rcParams["figure.figsize"] = [16, 16]
figure = plt.figure()
ax = plt.subplot(k, k, 1)
ax.set_title("Datos de entrada")
ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm.viridis, alpha=0.2, edgecolors='k') # entrenamiento
ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm.viridis, alpha=0.2, edgecolors='k') # validación
ax.set_xlim(xx.min(), xx.max())
ax.set_ylim(yy.min(), yy.max())
ax.set_xticks(())
ax.set_yticks(())
i = 2
for name, clf in zip(names, classifiers):
    ax = plt.subplot(k, k, i)
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    if hasattr(clf, "decision_function"):
        Z = clf.decision_function(c_[xx.ravel(), yy.ravel()])[:, 1]
    else:
        Z = clf.predict_proba(c_[xx.ravel(), yy.ravel()])[:, 1]
    Z = Z.reshape(xx.shape)
    ax.contourf(xx, yy, Z, cmap=cm.viridis, alpha=.8)
    ax.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=cm.viridis, edgecolors='k')
    ax.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=cm.viridis, edgecolors='k', alpha=0.6)
    ax.set_xlim(xx.min(), xx.max())
    ax.set_ylim(yy.min(), yy.max())
    ax.set_xticks(())
    ax.set_yticks(())
    ax.set_title(name)
    ax.text(xx.max() - .3, yy.min() + .3, ('%.3f' % score).lstrip('0'), size=40, horizontalalignment='right')
    i += 1
plt.tight_layout()
plt.show()
```





In [115]:
```
from sklearn import metrics
names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
    SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
    DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max
_features=1), \
    AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42) # la mis
ma división
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name, clf.score(X_test, y_test))
    expected, predicted = y_test, clf.predict(X_test)
    print(metrics.classification_report(expected, predicted))
    print(metrics.confusion_matrix(expected, predicted))
    print('-' * 60)
```

```
Nearest Neighbors 0.25
             precision    recall  f1-score   support

          0       0.33      0.60      0.43         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1

avg / total       0.14      0.25      0.18        12

[[3 1 0 0 0 0 1 0]
 [2 0 0 0 0 0 0 0]
```

```
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]]
------------------------------------------------------------
Linear SVM 0.3333333333333333
             precision    recall  f1-score   support

          0       0.44      0.80      0.57         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.19      0.33      0.24        12

[[4 1 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0]]
------------------------------------------------------------
RBF SVM 0.25
             precision    recall  f1-score   support

          0       0.33      0.60      0.43         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.14      0.25      0.18        12

[[3 2 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]]
------------------------------------------------------------

C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)

Gaussian Process 0.16666666666666666
             precision    recall  f1-score   support

          0       0.25      0.40      0.31         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1
```

```
          9       0.00      0.00      0.00         0

avg / total       0.10      0.17      0.13        12

[[2 2 0 0 0 0 0 1]
 [2 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
------------------------------------------------------------
Decision Tree 0.08333333333333333
             precision    recall  f1-score   support

          0       0.25      0.20      0.22         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1
          9       0.00      0.00      0.00         0

avg / total       0.10      0.08      0.09        12

[[1 1 0 0 0 1 1 0 1]
 [1 0 0 0 0 0 1 0 0]
 [0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0]]
------------------------------------------------------------
Random Forest 0.25
             precision    recall  f1-score   support

          0       0.40      0.40      0.40         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       1.00      1.00      1.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1
          9       0.00      0.00      0.00         0

avg / total       0.25      0.25      0.25        12

[[2 1 0 0 0 0 1 0 1]
 [2 0 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0]]
------------------------------------------------------------
AdaBoost 0.25
             precision    recall  f1-score   support

          0       0.33      0.60      0.43         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
```

```
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1
          9       0.00      0.00      0.00         0

avg / total       0.14      0.25      0.18        12

[[3 1 0 0 0 0 0 1]
 [2 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]]
------------------------------------------------------------
Naive Bayes 0.16666666666666666
             precision    recall  f1-score   support

          0       0.33      0.40      0.36         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.14      0.17      0.15        12

[[2 2 0 0 0 1 0]
 [2 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0]]
------------------------------------------------------------
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

Es muy baja la certeza por clasificar las enfermedades con los dos componentes obtenidos por PCA. Existen otros métodos como el del umbral de varianza (VT por sus siglas en inglés). Primero se escalan con base en una escala que no iguale la varianza de los datos

In [116]:
```
from sklearn.preprocessing import MinMaxScaler
x = MinMaxScaler().fit(x_copy).transform(x_copy)
```

para posteriormente seleccionar las características que tengan varianzas que pasen el umbral dado

In [117]:
```
# https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.VarianceThreshold.html#sklearn.feature_selection.VarianceThreshold
from sklearn.feature_selection import VarianceThreshold
th = .9 * (1 - .9)
print("Umbral de varianza = {}".format(th))
sel = VarianceThreshold(threshold = th)

sel.fit_transform( x )

xVT = x_copy[x_copy.columns[sel.get_support(indices=True)]]

xVT
```

```
Umbral de varianza = 0.08999999999999998
```

Out[117]:

	ac1	ac2	ac3	ac4	ac48
0	0.393749	0.122484	0.010328	0.026661	-0.017822
1	0.831565	0.690654	0.594390	0.516446	-0.016753
2	0.667701	0.505751	0.436771	0.352219	0.179380
3	0.620012	0.595386	0.535631	0.462663	0.284554
4	0.605797	0.479571	0.410353	0.367218	0.191535
5	0.380632	0.195406	0.107774	0.091505	0.126218
6	0.085993	0.048897	0.064574	0.118768	0.015544
7	0.522652	0.363609	0.361372	0.385651	0.235813
8	0.393243	0.260146	0.044702	0.032172	-0.020010
9	-0.007522	-0.007561	-0.007600	-0.007639	-0.000644
10	-0.081445	0.057375	-0.049496	0.012841	-0.031916
11	0.089659	-0.019229	-0.025867	-0.002608	0.005434
12	0.140500	0.121196	0.127669	0.099854	0.060802
13	0.003941	-0.012528	-0.008077	-0.013007	0.000458
14	0.076912	0.149122	0.088945	0.072881	-0.039804
15	0.895262	0.842421	0.792705	0.707147	0.466082
16	0.946239	0.892301	0.837622	0.778140	0.623759
17	0.488103	0.474707	0.426456	0.414057	0.087383
18	0.765747	0.587239	0.516973	0.418962	0.237261
19	0.289203	0.226359	0.276856	0.274657	0.018644
20	0.299805	0.225857	0.068495	0.057942	0.021663
21	0.149215	0.272471	0.152501	0.089044	0.013137
22	0.564047	0.398251	0.335638	0.284887	0.185766
23	0.767118	0.547797	0.395471	0.291686	-0.058954
24	0.077117	0.322187	0.046591	0.036037	-0.023882
25	0.388003	0.416624	0.313638	0.387392	0.207482
26	0.110847	0.069672	0.107836	0.080559	0.008848

27	0.232941	0.165279	0.155966	0.155952	-0.018886
28	0.853479	0.727355	0.620092	0.535945	0.204222
29	0.927733	0.858740	0.790416	0.727838	0.647336
30	0.055952	0.062823	0.076371	0.006623	-0.057666
31	0.791010	0.750958	0.691090	0.610929	0.511489
32	0.903920	0.826842	0.732156	0.614444	-0.008971
33	0.881185	0.793629	0.694008	0.596382	0.527848
34	0.026306	0.067629	0.068971	0.106583	0.006181
35	0.550097	0.512561	0.454005	0.411662	0.015187
36	0.844756	0.768593	0.723830	0.666654	0.521726
37	0.826727	0.781848	0.727522	0.675235	0.631255
38	0.497210	0.371337	0.338770	0.295026	0.020609

Ahora ya podrían analizarse tras escalarse con la escala estándar

In [118]:
```
X = StandardScaler().fit_transform(xVT)

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
    SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
    DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max
_features=1), \
    AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42) # la mis
ma división
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name, clf.score(X_test, y_test))
    expected, predicted = y_test, clf.predict(X_test)
    print(metrics.classification_report(expected, predicted))
    print(metrics.confusion_matrix(expected, predicted))
    print('-' * 60)
```

```
Nearest Neighbors 0.4166666666666667
             precision    recall  f1-score   support

          0       0.57      0.80      0.67         5
          1       0.33      0.50      0.40         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.29      0.42      0.34        12

[[4 1 0 0 0 0 0 0]
 [1 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 0 1 0 0 0]]
------------------------------------------------------------
Linear SVM 0.4166666666666667
             precision    recall  f1-score   support

          0       0.42      1.00      0.59         5
```

```
          0       0.42      1.00      0.59         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.17      0.42      0.25        12

[[5 0 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]]
------------------------------------------------------------
RBF SVM 0.4166666666666667
             precision    recall  f1-score   support

          0       0.44      0.80      0.57         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.50      1.00      0.67         1
          8       0.00      0.00      0.00         1

avg / total       0.23      0.42      0.29        12

[[4 1 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [0 0 0 0 0 1 0]
 [0 0 0 0 0 1 0]]
------------------------------------------------------------
Gaussian Process 0.3333333333333333
             precision    recall  f1-score   support

          0       0.44      0.80      0.57         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.19      0.33      0.24        12

[[4 1 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0]]
------------------------------------------------------------
Decision Tree 0.25
             precision    recall  f1-score   support

          0       0.50      0.40      0.44         5
          1       0.50      0.50      0.50         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
```

```
          6       0.00      0.00      0.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1
         10       0.00      0.00      0.00         0

avg / total       0.29      0.25      0.27        12

[[2 1 0 0 1 0 0 1 0 0]
 [0 1 0 0 0 0 0 0 0 1]
 [1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0 0]]
------------------------------------------------------------
Random Forest 0.3333333333333333
             precision    recall  f1-score   support

          0       0.40      0.40      0.40         5
          1       0.50      0.50      0.50         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       1.00      1.00      1.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1
          9       0.00      0.00      0.00         0

avg / total       0.33      0.33      0.33        12

[[2 1 0 0 0 0 1 0 1]
 [1 1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 0 0 0]]
------------------------------------------------------------
AdaBoost 0.4166666666666667
             precision    recall  f1-score   support

          0       0.50      0.60      0.55         5
          1       0.67      1.00      0.80         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.32      0.42      0.36        12

[[3 1 0 0 1 0 0 0]
 [0 2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 0 1 0 0 0]]
------------------------------------------------------------
Naive Bayes 0.25
             precision    recall  f1-score   support

          0       0.40      0.40      0.40         5
```

```
          0       0.40      0.40      0.40         5
          1       0.50      0.50      0.50         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1

avg / total       0.25      0.25      0.25        12

[[2 1 0 0 1 0 0 1 0]
 [1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]

 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]]
------------------------------------------------------------
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

Aunque no logran clasificar decentemente, algunos algoritmos mejoran un poco, por lo que se intentará clasificar nuevamente pero escalando con el algoritmo MinMax

In [119]:

```
X = MinMaxScaler().fit(xVT).transform(xVT)

names = ["Nearest Neighbors", "Linear SVM", "RBF SVM", "Gaussian Process", \
         "Decision Tree", "Random Forest", "AdaBoost", "Naive Bayes"]
classifiers = [KNeighborsClassifier(3), SVC(kernel="linear", C=0.025), \
    SVC(gamma=2, C=1), GaussianProcessClassifier(1.0 * RBF(1.0)), \
    DecisionTreeClassifier(max_depth=5), RandomForestClassifier(max_depth=5, n_estimators=10, max
_features=1), \
    AdaBoostClassifier(), GaussianNB()]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.3, random_state=42) # la mis
ma división
for name, clf in zip(names, classifiers):
    clf.fit(X_train, y_train)
    print(name, clf.score(X_test, y_test))
    expected, predicted = y_test, clf.predict(X_test)
    print(metrics.classification_report(expected, predicted))
```

```
    print(metrics.confusion_matrix(expected, predicted))
    print('-' * 60)
```

```
Nearest Neighbors 0.4166666666666667
             precision    recall  f1-score   support

          0       0.57      0.80      0.67         5
          1       0.33      0.50      0.40         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.29      0.42      0.34        12

[[4 1 0 0 0 0 0 0]
 [1 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 1 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 0 1 0 0 0]]
------------------------------------------------------------
Linear SVM 0.4166666666666667
             precision    recall  f1-score   support

          0       0.42      1.00      0.59         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.17      0.42      0.25        12

[[5 0 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]]
------------------------------------------------------------
RBF SVM 0.4166666666666667
             precision    recall  f1-score   support

          0       0.50      0.80      0.62         5
          1       0.25      0.50      0.33         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.25      0.42      0.31        12

[[4 1 0 0 0 0 0]
 [1 1 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0]]
------------------------------------------------------------
Gaussian Process 0.3333333333333333
             precision    recall  f1-score   support
```

```
          0       0.44      0.80      0.57         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.19      0.33      0.24        12

[[4 1 0 0 0 0 0]
 [2 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0]
 [0 1 0 0 0 0 0]
 [0 1 0 0 0 0 0]]
------------------------------------------------------------
Decision Tree 0.3333333333333333
             precision    recall  f1-score   support

          0       0.40      0.40      0.40         5
          1       0.50      0.50      0.50         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       1.00      1.00      1.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1

avg / total       0.33      0.33      0.33        12

[[2 1 0 0 1 0 0 1 0]
 [1 1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0]
 [0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]]
------------------------------------------------------------
Random Forest 0.3333333333333333
             precision    recall  f1-score   support

          0       0.38      0.60      0.46         5
          1       0.00      0.00      0.00         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.50      1.00      0.67         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1

avg / total       0.20      0.33      0.25        12

[[3 0 0 0 1 0 0 1 0]
 [2 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 1 0 0]]
------------------------------------------------------------
```

```
AdaBoost 0.3333333333333333
             precision    recall  f1-score   support

          0       0.43      0.60      0.50         5
          1       0.50      0.50      0.50         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          8       0.00      0.00      0.00         1

avg / total       0.26      0.33      0.29        12

[[3 1 0 0 1 0 0 0]
 [1 1 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0]
 [0 0 0 0 1 0 0 0]]
------------------------------------------------------------
Naive Bayes 0.25
             precision    recall  f1-score   support

          0       0.40      0.40      0.40         5
          1       0.50      0.50      0.50         2
          2       0.00      0.00      0.00         1
          3       0.00      0.00      0.00         1
          4       0.00      0.00      0.00         0
          5       0.00      0.00      0.00         1
          6       0.00      0.00      0.00         1
          7       0.00      0.00      0.00         0
          8       0.00      0.00      0.00         1

avg / total       0.25      0.25      0.25        12

[[2 1 0 0 1 0 0 1 0]
 [1 1 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 1 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [1 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]
 [0 0 0 0 0 0 0 0 0]
 [0 0 0 0 1 0 0 0 0]]
------------------------------------------------------------

C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)

C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1135: UndefinedMetricWarning: Precision and F-score are ill-defined and being
 set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)
```

```
  'precision', 'predicted', average, warn_for)
C:\Users\bena8\AppData\Local\Programs\Python\Python37-32\lib\site-packages\sklearn\metrics\cl
assification.py:1137: UndefinedMetricWarning: Recall and F-score are ill-defined and being se
t to 0.0 in labels with no true samples.
  'recall', 'true', average, warn_for)
```

El RandomForest mejora mientras el AdaBoost empeora y los demás se quedan igual, así que la escala estándar sería preferente para AdaBoost mientras que la MinMax con rango $[-1, 1]$ es mejor para RandomForest.

Ciencia de datos

Práctica 11. Clasificación de datos con sklearn

Alberto Benavides

Para esta práctica se parte de los datos cuyas características fueron seleccionadas por el algoritmo del umbral de varianza, a saber

In [91]:
```
import pandas as pd

x = pd.read_csv("/mnt/c/Users/bena8/Documents/FIME/Tesis/Resultados/xVT.csv")
x.sample(3)
```

Out[91]:

	ac1	ac2	ac3	ac4	ac48
15	0.895262	0.842421	0.792705	0.707147	0.466082
32	0.903920	0.826842	0.732156	0.614444	-0.008971
10	-0.081445	0.057375	-0.049496	0.012841	-0.031916

In [92]:
```
y = pd.read_csv("/mnt/c/Users/bena8/Documents/FIME/Tesis/Resultados/y.csv")
y.sample(3)
```

Out[92]:

	cie
10	a34
38	z21
6	a17.0

Asimismo, se cargan las CIEs correspondientes a cada registro

In [94]:
```
y = pd.read_csv("/mnt/c/Users/bena8/Documents/FIME/Tesis/Resultados/y.csv")
y.sample(3)
```

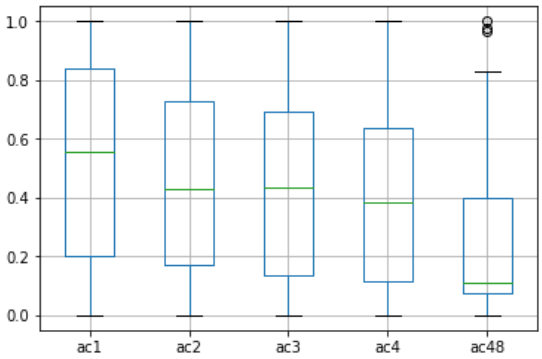
Out[94]:

	cie
26	g00-g03
2	a01.0
8	a27

Las características seleccionadas corresponden a las autocorrelaciones con retrasos de 1, 2, 3, 4 y 48 semanas. Es importante primero normalizar las características que serán la entrada de los algoritmos de agrupamiento

In [95]:
```
from sklearn.preprocessing import MinMaxScaler
X = MinMaxScaler().fit(x).transform(x)

import matplotlib.pyplot as plt

# https://stackoverflow.com/a/20763459
X = pd.DataFrame(data=X)
# https://stackoverflow.com/a/11346337
X.columns = x.columns
X.boxplot()
```

Out[95]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6ef8653860>
```



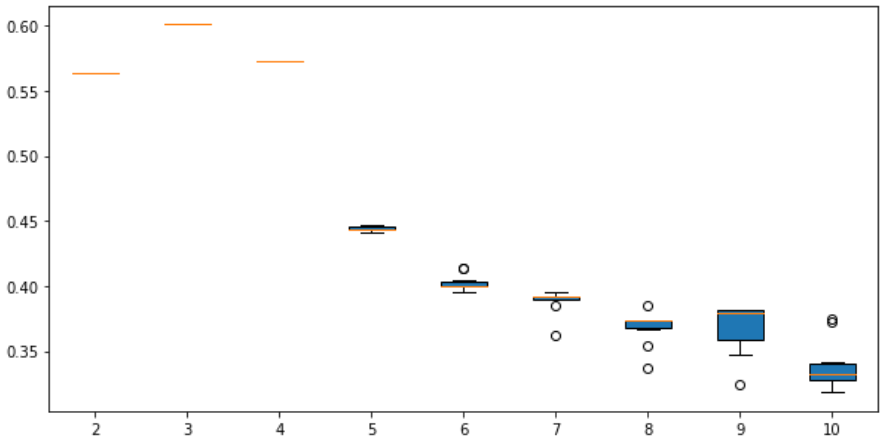
De las CIEs, nos interesan las CIEs generales, o sea la primera letra de cada CIE

In [96]:
```
cieG = y.cie.str[0]

y.cie.str[0].unique()
```

Out[96]:
```
array(['a', 'b', 'g', 'i', 'j', 'p', 't', 'u', 'w', 'x', 'z'],
      dtype=object)
```

Con base en esto, se utilizan dos algoritmos para realizar el agrupamiento. Priero k-medias, por ser el algoritmo más utilizado. Con este algoritmo es necesario especificar el número de agrupamientos. Para ello, se utilizará el método del codo

In [121]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
import numpy as np

clusters = range(2, len(cieG.unique()) + 1)
l = len(clusters) - 1
results_test = np.zeros(shape=(0, l))
for j in range(10):
    kmeans = [KMeans(n_clusters=i) for i in clusters]
    y_pred_test = [kmeans[i].fit(X).predict(X) for i in range(len(kmeans))]
    score = [silhouette_score(X, y_pred_test[i]) for i in range(len(kmeans) - 1)]
    results_test = np.vstack((results_test, score))

fig = plt.figure(figsize=(10, 5))
ax = fig.add_subplot(111)
ax.boxplot(results_test, patch_artist=True)
ax.set_xticklabels(clusters) # https://matplotlib.org/gallery/statistics/boxplot_demo.html
plt.show()

from kneed import KneeLocator
kneedle = KneeLocator(
            range(2, len(cieG.unique())), # Entre los clusters definidos
            np.mean(results_test, axis=0),
            curve='convex', direction='increasing')
# Se actualiza el número de clusters
n_clusters = kneedle.knee
print("Clusters elegidos: {}\n". format(n_clusters))
```




```
Clusters elegidos: 3
```

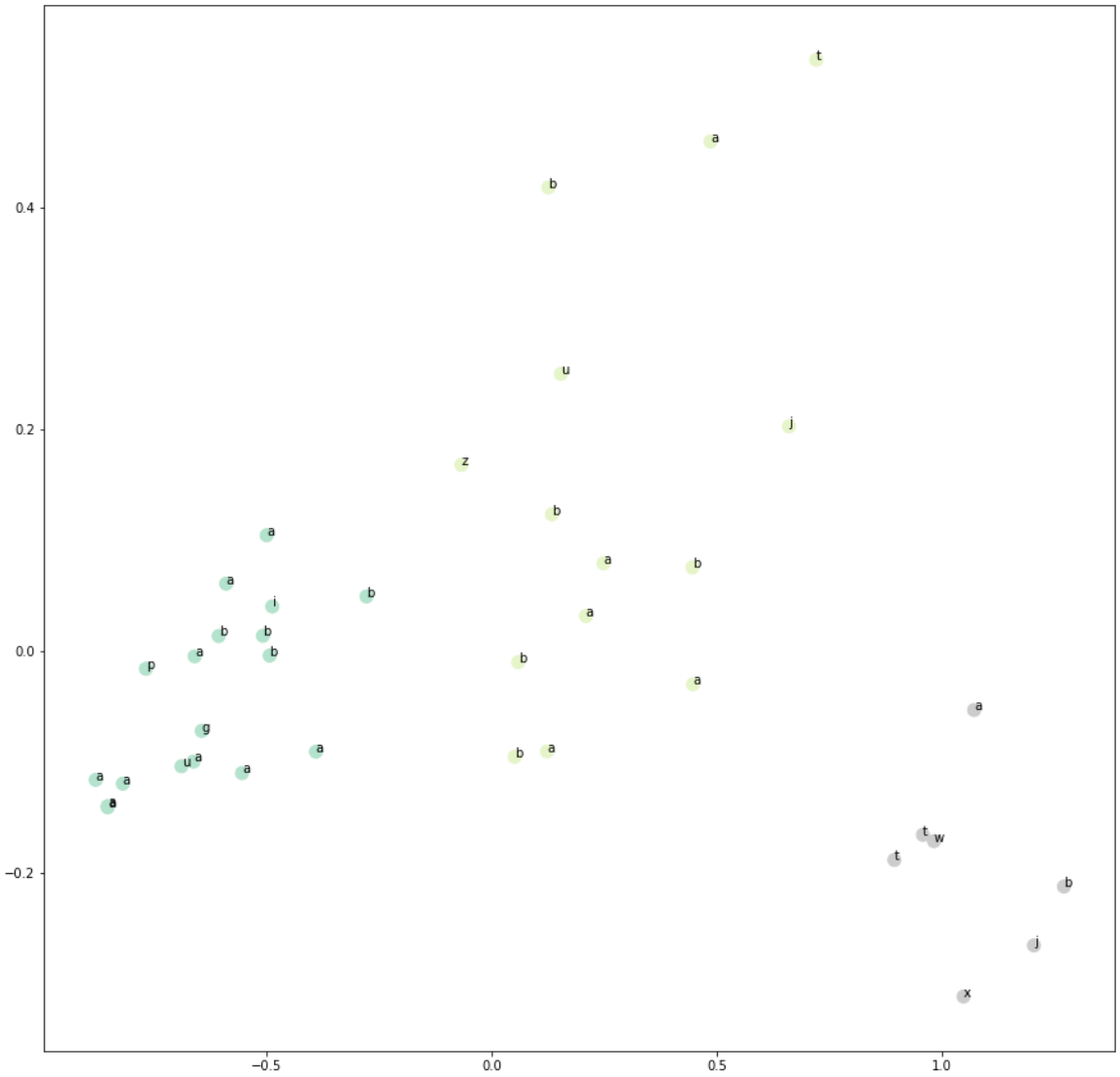
Con este conocimiento, se puede ejecutar k-medias con $k = 6$ para este conjunto de datos y obtener los grupos

In [130]:
```
y_pred = KMeans(n_clusters=3).fit(X).predict(X)
y_pred
```

Out[130]:
```
array([0, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 1, 0, 0, 0,
       1, 1, 0, 1, 0, 0, 1, 2, 0, 2, 1, 2, 0, 1, 2, 2, 1], dtype=int32)
```

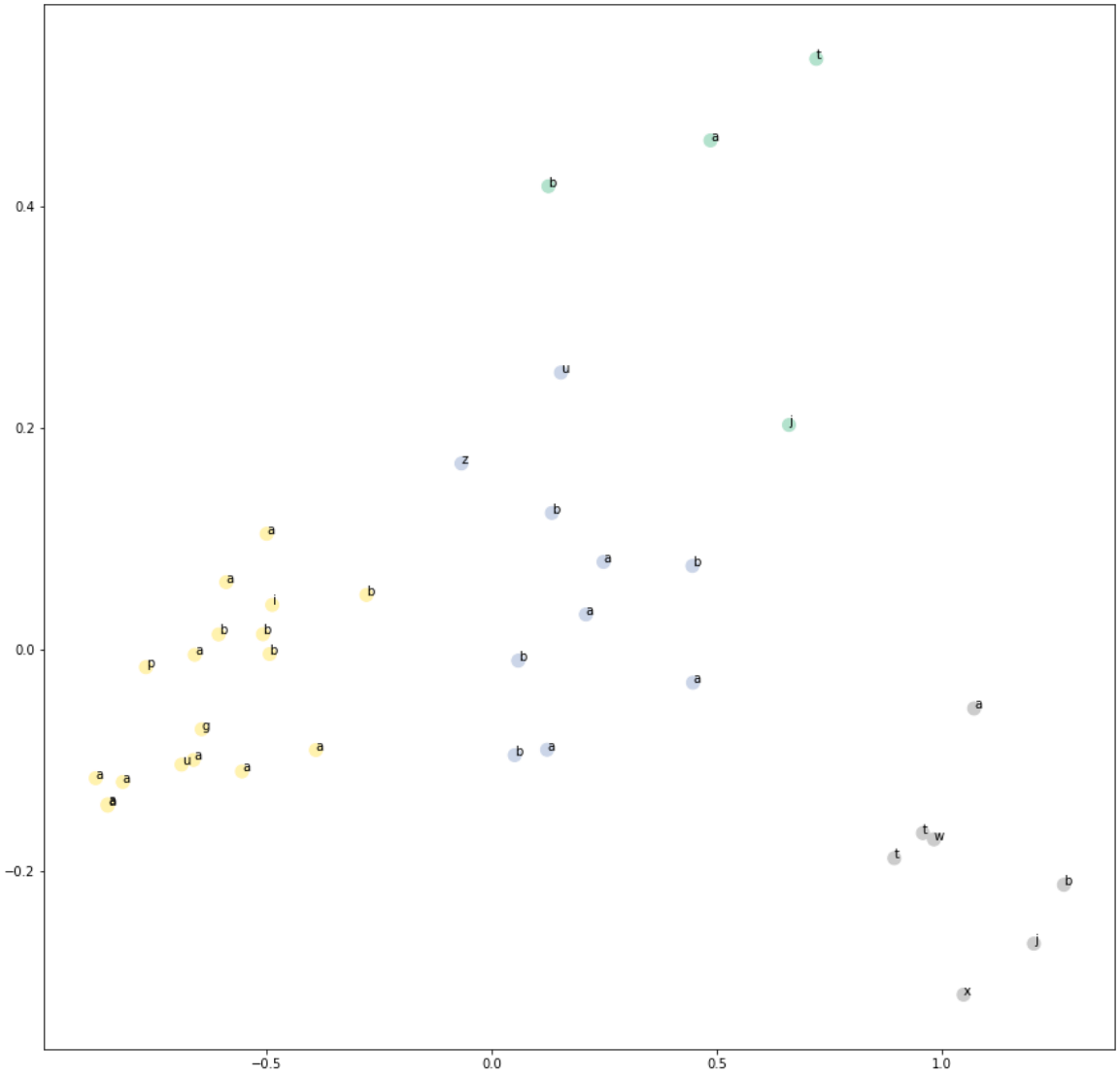
Estos valores se pueden mostrar en una gráfica

In [131]:
```
from sklearn.decomposition import PCA
from matplotlib import cm as cm

cmap = cm.get_cmap('Pastel2', 30)

pca = PCA(n_components = 2).fit_transform(X)
pca = pd.DataFrame(data=pca)

fig, ax = plt.subplots(figsize=(15,15))
ax.scatter(pca[0], pca[1], c=y_pred, s = 100, cmap = cmap)
for i, txt in enumerate(y.cie.str[0]):
    ax.annotate(txt, (pca[0][i], pca[1][i]))

plt.show()
```



Mas se nota que el agrupamiento es más al azar que nada. Así que se probará con otro método que quizás sea más adecuado para estas series de tiempo, que es el agrupamiento de Affinity propagation

In [132]:
```
from sklearn.cluster import AffinityPropagation


y_pred_ap = AffinityPropagation().fit(X).predict(X)

fig, ax = plt.subplots(figsize=(15,15))
ax.scatter(pca[0], pca[1], c=y_pred_ap, s = 100, cmap= cmap)
for i, txt in enumerate(y.cie.str[0]):
    ax.annotate(txt, (pca[0][i], pca[1][i]))

plt.show()

print(silhouette_score(X, y_pred_ap, metric='euclidean'))
```

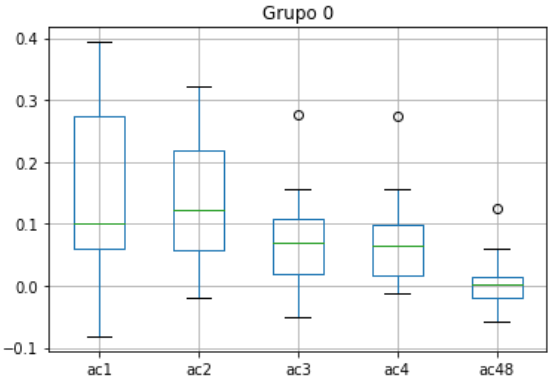


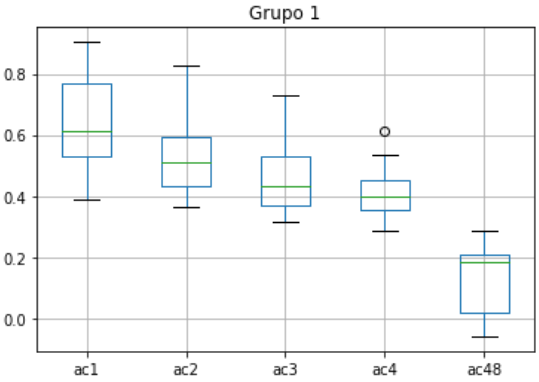

```
0.5593729217401119
```

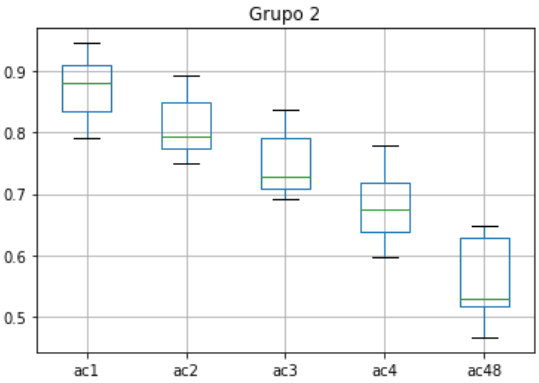
Aquí aparecen menos dispersas, pero bien podría ser debido a que hay más grupos que se forman, aún así no mejora la medida del error. Pese a todo esto, es posible mostrar las características de estos agrupamientos por separado para compararlas

In [180]:
```
t = pd.DataFrame(data=y_pred)
```

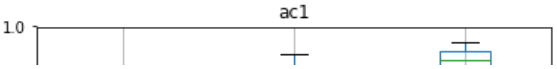
```
all = x.join(t)
all = all.rename(columns = {0:'y'})

for n, g in all.groupby('y'):
    plt.figure()
    g.iloc[:, :-1].boxplot()
    plt.title("Grupo " + str(n))
    plt.show()

for n in all.iloc[:, :-1]:
    plt.figure()
    all.boxplot(column=[n], by='y')
    plt.suptitle('')
    plt.title(n)
    plt.xlabel('Grupo')
    plt.show()
```

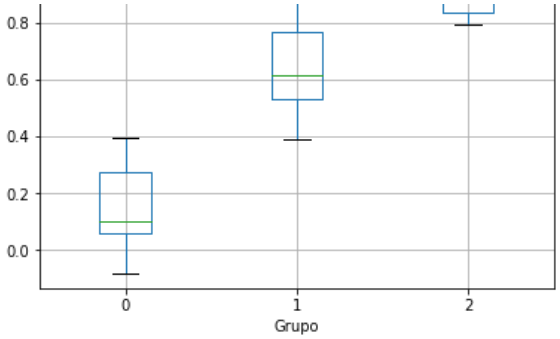




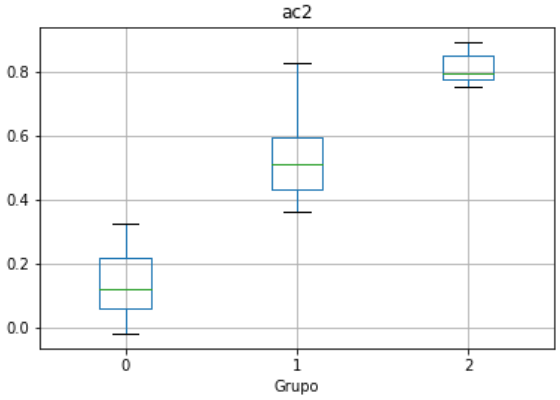



```
<Figure size 432x288 with 0 Axes>
```

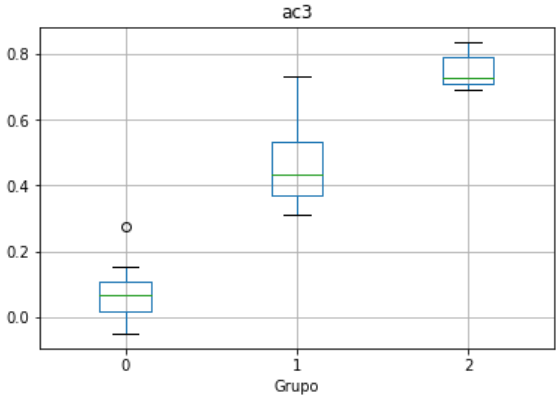




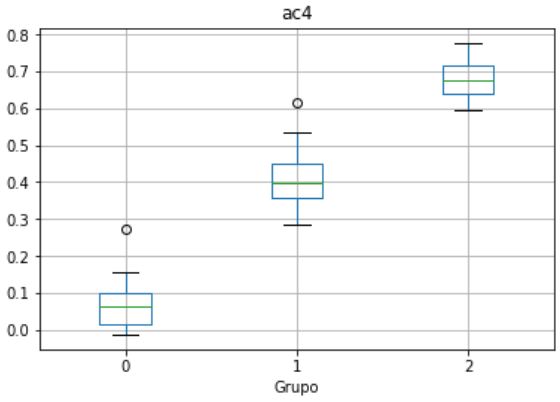

```
<Figure size 432x288 with 0 Axes>
```



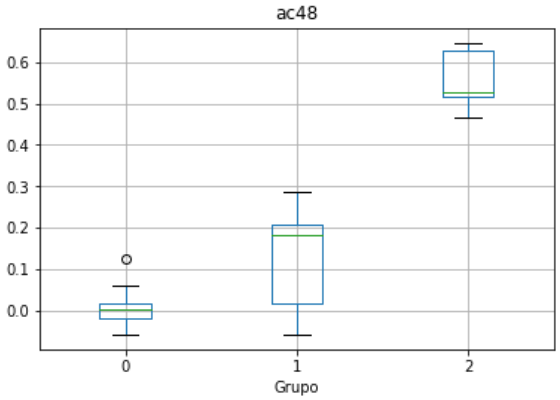

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 432x288 with 0 Axes>
```



```
<Figure size 432x288 with 0 Axes>
```



Esto muestra que las diferencias entre grupos se deben, principalmente, a que los grupos 0 y 1 tienen autocorrelaciones con retraso de 1, 2, 3 y 4 semansa distintas y que no se sobrelapan. Entre los grupos 1 y 2 hay un poco de sobrelapamiento en las autocorrelaciones mencionadas, pero se distinguen por la autocorrelación de 48 semanas.

Con esta información se puede concluir que los grupos generados no tienen relación con la CIE propuesta por la OMS, sin embargo es posible extraer información de interés respecto a las autocorrelaciones de estas series de tiempo.

Práctica 12: Análisis de textos

Alberto Benavides

En esta práctica se analiza la autoría de textos que se consideran espurios del filósofo Platón, uno de los primeros representantes de la filosofía idealista, famoso discípulo de Sócrates y maestro de Aristóteles. Actualmente algunas de las traducciones de sus obras se encuentran libres de derecho de autor y pueden descargarse del sitio Gutenberg Project (https://www.gutenberg.org/wiki/Main_Page).

Las obras elegidas (y disponibles) son:

- *Apología*,
- *Eutifrón*,
- *Gorgias*,
- *Leyes*,
- *Menón*,
- *Fedro*,
- *Protágoras*,
- *República*,
- *Alcibíades I*,
- *Alcibíades II*,
- *Erixias*,
- *Hipias menor*,
- *Simposio*,
- *Teeteto*, y
- *Timeo*.

María Gardella (https://www.teseopress.com/amantesrivales/chapter/1-sobre-el-corpus-de-dialogos-dudosos-y-apocrifos/) recopila algunas clasificaciones tradicionales y actuales sobre obras dudosas y apócrifas del autor, base sobre la que se catalogan las obras *Alcibíades I* y *II*, *Erixias* e *Hipias menor* como espurias.

A partir de estas traducciones, los textos se agruparán por k-medias siguiendo esta metodología (http://www.aicbt.com/authorship-attribution/) con base en la siguiente lista de características extraida de cada diálogo:

- media de palabras por oración,
- desviación estándar de palabras por oración,
- diversidad léxica (palabras sin repetir entre total de palabras utilizadas),
- comas (,) por oración,
- punto y comas (;) por oración,
- dos puntos (:) por oración,
- palabras negativas,
- palabras neutras, y
- palabras positivas.

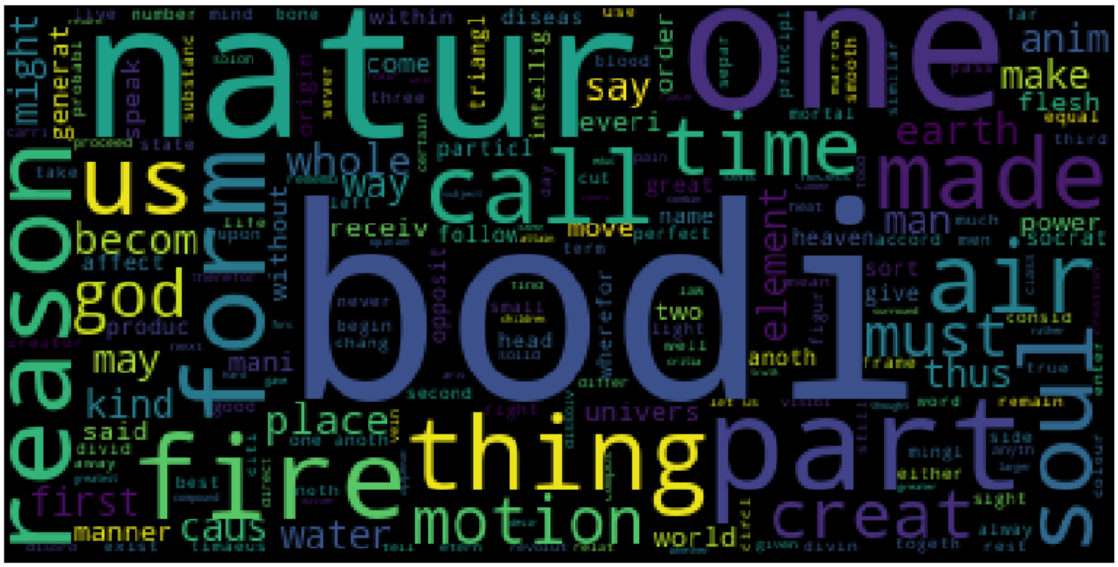
Es importante señalar que estos resultados son sólo una somera aproximación de este tipo de análisis, puesto que se trata de traducciones y no de las obras en griego. Afortunadamente, las obras en griego clásico se encuentran también libres de derechos de autor y pueden ser descargadas para su posterior análisis.

En cuanto al análisis, primero se extraen los textos completos, se eliminan artículos y preposiciones, se estandarizan las formas verbales, adjetivos y artículos, se extraen las características mencionadas arriba y se presentan dos nubes de palabras, una para el conjunto de textos considerados platónicos y otra para los espurios.

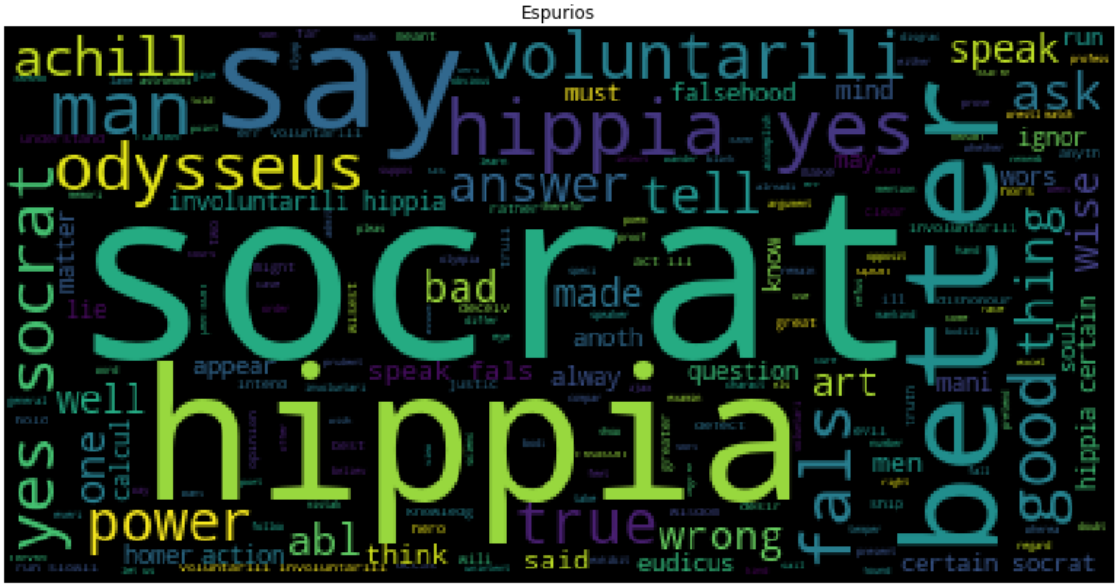
In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import SnowballStemmer
from nltk.tokenize import RegexpTokenizer
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import os
import nltk
import numpy as np
from nltk import word_tokenize

sentence_tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
word_tokenizer = nltk.tokenize.RegexpTokenizer(r'\w+')


dir = "/mnt/c/Users/bena8/Documents/FIME/Ciencia de datos/plato"

plato = ""
spu = ""

df = pd.DataFrame(columns=["wpsMean", "wpsStd", "lexDiv", "comps", "scolps", "colps", "neg", "neu
", "pos", "title"])

spa = stopwords.words('english')
stemmer = SnowballStemmer('english')
s = SentimentIntensityAnalyzer()

i = 0

for filename in os.listdir(dir):
    with open(dir + "/" + filename) as f:
        text = f.read()
        #text = spell(text)
        quedar = [stemmer.stem(p) for p in word_tokenizer.tokenize(text) if p.lower() not in spa]
        if filename.startswith("spu"):
            spu = " ".join(quedar)
        else:
            plato = " ".join(quedar)
        tokens = nltk.word_tokenize(text.lower())
        words = word_tokenizer.tokenize(text.lower())
        sentences = sentence_tokenizer.tokenize(text)
        vocab = set(words)
        words_per_sentence = np.array([len(word_tokenizer.tokenize(s))
                                       for s in sentences])
        pol = s.polarity_scores(text)
        df.loc[i] = [
            words_per_sentence.mean(),
            words_per_sentence.std(),
            len(vocab) / float(len(words)),
            tokens.count(',') / float(len(sentences)),
            tokens.count(';') / float(len(sentences)),
            tokens.count(':') / float(len(sentences)),
            pol.get('neg'), pol.get('neu'), pol.get('pos'),
            filename
        ]
        i += 1

nubePlato = WordCloud().generate(plato)
nubeSpu = WordCloud().generate(spu)
```

In [6]:
```
plt.rcParams["figure.figsize"] = [15, 7]
plt.title("Platónicos")
plt.imshow(nubePlato)
plt.axis("off")
plt.show()

plt.rcParams["figure.figsize"] = [15, 7]
plt.title("Espurios")
plt.imshow(nubeSpu)
plt.axis("off")
plt.show()
```

Platónicos



Espurios



Aparentemente, los diálogos espurios contienen muchas intervenciones de Sócrates e Hipias, mientras que los considerados generalmente platónicos contienen referencias hacia la naturaleza y el cuerpo (aunque con errores ortográficos con los que se tendría que lidiar en futuras investigaciones).

Enseguida se muestra un cuadro con las características obtenidas por diálogo.

In [2]:
```
df
```

Out[2]:

	wpsMean	wpsStd	lexDiv	comps	scolps	colps	neg	neu	pos	title
0	29.664083	25.233705	0.147387	2.312661	0.441860	0.144703	0.124	0.727	0.149	apology.txt
1	18.063830	14.418191	0.146790	1.321809	0.212766	0.667553	0.103	0.734	0.163	euthyphro.txt
2	21.817302	23.823621	0.083851	1.773140	0.251664	0.754991	0.117	0.701	0.182	gorias.txt
3	34.066811	29.952127	0.052237	2.550651	0.435118	0.504342	0.096	0.760	0.144	laws.txt
4	16.740885	16.028354	0.110368	1.190104	0.201823	0.809896	0.065	0.745	0.190	meno.txt
5	29.889318	28.024867	0.139769	2.077220	0.486486	0.544402	0.080	0.731	0.189	phaedrus.txt
6	25.097056	24.212760	0.110846	2.114504	0.365322	0.173391	0.104	0.733	0.163	protagoras.txt
7	22.076895	23.847615	0.061511	1.700204	0.295164	0.058180	0.076	0.726	0.197	republic.txt
8	14.216891	15.178433	0.112596	0.966411	0.129559	0.880038	0.068	0.735	0.197	spu_alcibiadesI.txt
9	19.010309	17.693163	0.197216	1.357388	0.082474	0.731959	0.099	0.745	0.155	spu_alcibiadesII.txt
10	22.151724	19.200334	0.167186	1.465517	0.110345	0.582759	0.078	0.696	0.226	spu_eryxias.txt
11	20.023179	21.415693	0.159087	1.576159	0.248344	0.791391	0.119	0.737	0.144	spu_lesserHipias.txt
12	30.790055	24.956263	0.129553	2.602210	0.484807	0.135359	0.061	0.702	0.236	symposium.txt
13	22.350852	23.630301	0.096727	1.785511	0.307528	0.793324	0.069	0.798	0.133	theaetetus.txt
14	43.948856	29.657172	0.113248	3.414536	0.547779	0.134590	0.061	0.817	0.122	timaeus.txt

Y, finalmente, se agrupan por k-medias con base en las características descritas. Este algoritmo requiere que se especifique el número k de grupos en que se dividirán los diálogos. En este caso se ha elegido una $k = 2$ puesto que se esperaría tener dos grupos contuvieran, uno, textos platónicos y, el otro, espurios.

In [4]:
```
from sklearn.cluster import KMeans
from scipy.cluster.vq import whiten

df_copy = df

x = df_copy.iloc[:,:-2]
y = df_copy.iloc[:,-1]

x = whiten(x.values)

km = KMeans(n_clusters=2, init='k-means++', n_init=10, verbose=0).fit(x)

df['group'] = km.labels_
```

In [7]:
```
df
```

Out[7]:

	wpsMean	wpsStd	lexDiv	comps	scolps	colps	neg	neu	pos	title	group
0	29.664083	25.233705	0.147387	2.312661	0.441860	0.144703	0.124	0.727	0.149	apology.txt	0
1	18.063830	14.418191	0.146790	1.321809	0.212766	0.667553	0.103	0.734	0.163	euthyphro.txt	1
2	21.817302	23.823621	0.083851	1.773140	0.251664	0.754991	0.117	0.701	0.182	gorias.txt	1
3	34.066811	29.952127	0.052237	2.550651	0.435118	0.504342	0.096	0.760	0.144	laws.txt	0
4	16.740885	16.028354	0.110368	1.190104	0.201823	0.809896	0.065	0.745	0.190	meno.txt	1
5	29.889318	28.024867	0.139769	2.077220	0.486486	0.544402	0.080	0.731	0.189	phaedrus.txt	0
6	25.097056	24.212760	0.110846	2.114504	0.365322	0.173391	0.104	0.733	0.163	protagoras.txt	0
7	22.076895	23.847615	0.061511	1.700204	0.295164	0.058180	0.076	0.726	0.197	republic.txt	0
8	14.216891	15.178433	0.112596	0.966411	0.129559	0.880038	0.068	0.735	0.197	spu_alcibiadesI.txt	1
9	19.010309	17.693163	0.197216	1.357388	0.082474	0.731959	0.099	0.745	0.155	spu_alcibiadesII.txt	1
10	22.151724	19.200334	0.167186	1.465517	0.110345	0.582759	0.078	0.696	0.226	spu_eryxias.txt	1
11	20.023179	21.415693	0.159087	1.576159	0.248344	0.791391	0.119	0.737	0.144	spu_lesserHipias.txt	1
12	30.790055	24.956263	0.129553	2.602210	0.484807	0.135359	0.061	0.702	0.236	symposium.txt	0
13	22.350852	23.630301	0.096727	1.785511	0.307528	0.793324	0.069	0.798	0.133	theaetetus.txt	1
14	43.948856	29.657172	0.113248	3.414536	0.547779	0.134590	0.061	0.817	0.122	timaeus.txt	0

Los resultados arrojados por k-medias efectivamente agrupan los diálogos que se consideran espurios juntos, sin embargo incluyen también: *Eutifrón*, *Gorgias*, *Menón* y *Teeteto* con base en las características y traducciones elegidas.

In [ ]:
```
```