Contagios diarios COVID-19 en Colombia

Daily contagions COVID-19 in Colombia

Autor: Luis Fernando Zuluaga Torres – Jaime Andrés Marín Alarcón

*IS&C, Universidad Tecnológica de Pereira, Pereira, Colombia*

Correo-e: [Luis.zuluaga1@utp.edu.co](mailto:Luis.zuluaga1@utp.edu.co) – [Jaime.marin@utp.edu.co](mailto:Jaime.marin@utp.edu.co)

***Resumen*— En este documento, se hace una recopilación de datos diarios, desde el 10 de marzo del 2020 hasta el 27 de septiembre del 2020, en los que se puede evidenciar el crecimiento de los nuevos casos de contagios del nuevo coronavirus, además, se hace una predicción aproximada del número de contagios esperados, para una determinada fecha.**

***Palabras clave—* contagios, crecimiento, datos, dispersión, análisis, investigación, funciones, librería, calculo, predicción.**

***Abstract*— In this document, a collection of daily data is made, from March 10, 2020 to September 27, 2020, in which the growth of new cases of new coronavirus infections can be evidenced, in addition, a approximate prediction of the number of infections expected, for a certain date.**

***Key Word*— contagions, growth, data, dispersion, analysis, research, functions, library, calculation, prediction.**

1. INTRODUCCIÓN

El SARS-CoV -2, conocido como covid 19, es un virus que a azotado todo el mundo, causando aproximadamente 1 millón de muertes, y superando más de los 33 millones de contagios a nivel mundial. En nuestro país, que está ubicado actualmente en el 5to puesto de contagios a nivel global con más de 800 mil casos registrados, y más de 25 mil muertes, se vive actualmente una crisis, tanto sanitaria como económica debido a la poca concientización de la gente.

A continuación, se realizará un análisis de las cifras de nuevos contagios por cada día, desde el 10 de marzo hasta la actualidad, implementándolos, y operándolos en el notebook de jupyter, pudiendo mostrar gráficamente, el comportamiento del crecimiento de la curva de contagios.

* 1. RECOPILACION DE DATOS

Como se ha mencionado anteriormente, se han elegido los datos desde el 10 de marzo hasta el día de hoy, completando así, una cantidad de 202 datos, que vendrían siendo aproximadamente unos 6 meses, evidenciando las cifras diarias de contagios.

|  |
| --- |
| 1 0  2 6  3 4  4 3  5 8  6 21  7 12  8 18  9 27  10 26  11 47  12 35  13 30  14 66  15 113  16 62  17 10  18 48  19 69  20 94  21 96  22 108  23 159  24 96  25 106  26 106  27 79  28 94  29 201  30 274  31 169  32 250  33 236  34 67  35 76  36 127  37 126  38 128  39 206  40 182  41 171  42 185  43 172 |

|  |
| --- |
| 44 207  45 205  46 320  47 261  48 237  49 218  50 352  51 262  52 296  53 499  54 279  55 383  56 305  57 640  58 346  59 497  60 595  61 444  62 568  63 550  64 659  65 658  66 680  67 606  68 723  69 635  70 721  71 640  72 752  73 643  74 801  75 1046  76 998  77 806  78 1022  79 1101  80 1262  81 1322  82 1548  83 1147 |

|  |
| --- |
| 84 1110  85 1340  86 1521  87 1766  88 1515  89 1932  90 1209  91 1483  92 1359  93 1604  94 1530  95 1646  96 1888  97 2193  98 2124  99 1868  100 2115  101 3171  102 3059  103 2357  104 3019  105 2531  106 2389  107 3541  108 3486  109 3843  110 4149  111 3178  112 3174  113 2803  114 4163  115 4101  116 3395  117 3884  118 3721  119 3171  120 4213  121 4144  122 5335  123 6803  124 4586  125 5083  126 3832  127 5621  128 5271 |

|  |  |  |
| --- | --- | --- |
| 129 8037  130 8934  131 8560  132 6578  133 6727  134 7033  135 7390  136 7945  137 7168  138 7254  139 8181  140 8125  141 10284  142 8670  143 9965  144 9488  145 10673  146 11470  147 10199  148 7129  149 10735  150 11996  151 9486  152 9674  153 1061  154 10142  155 12830  156 12066  157 11286  158 11306  159 11578  160 11643  161 8328  162 12462  163 13056  164 11541  165 8419  166 10965  167 8044  168 10549  169 10432 |  | 170 10142  171 9752  172 8498  173 9394  174 8024  175 7230  176 8901  177 9270  178 8235  179 8488  180 8394  181 8065  182 5327  183 7665  184 7338  185 7813  186 7424  187 6826  188 7355  189 5573  190 6698  191 7787  192 7568  193 6526  194 7927  195 6678  196 5359  197 7102  198 6730  199 6555  200 7494  201 7721  202 7018 |

* 1. IMPORTACION DE LIBRERIAS

La primera librería que se importa, es la del sistema operativo, seguida de la librería numpy, y la librería utils, que nos servirán para el procesamiento de datos y la carga del archivo de datos respectivamente.

import os

from utils import DATA\_DIR, CHART\_DIR

import numpy as np

Se importa la librería scipy y la librería matplotlib

Esta librería esta organiza por subpaquetes donde cada 1 está enfocado a un tema de cálculos específicos como en este

Caso que se trabajara con numpy

* 1. FUNCIONES Y CODIGO

Lo primero que se hace, es eliminar las advertencias, que se hace desde la librería numpy. Esto, ya que en el futuro las funciones cambiaran.

np.seterr(all='ignore')

se importa el archivo que contiene los datos recogidos.

data = np.genfromtxt(os.path.join(DATA\_DIR, "web\_traffic.tsv"),

delimiter="\t")

luego se establece el tipo de dato del archivo importado.

data = np.array(data, dtype=np.float64)

print(data[:10])

print(data.shape)

se definen los colores, y estilos de línea que se van a utilizar para las graficas.

colors = ['g', 'k', 'b', 'm', 'r']

linestyles = ['-', '-.', '--', ':', '-']

ademas se crea un vector para cada columna correspondiente para cada archive importado.

x = data[:, 0]

y = data[:, 1]

la función isnan(vector) devuelve un vector en el cual los TRUE

son valores de tipo nan, y los valores FALSE son valores diferentes

a nan. Con esta información, este vector permite realizar

transformaciones a otros vectores (o al mismo vector), y realizar

operaciones como sumar el número de posiciones TRUE, con lo

cual se calcula el total de valores tipo nan

print("Número de entradas incorrectas:", np.sum(np.isnan(y)))

con la siguiente función, podemos eliminar los valores nan,

se crea un vector TRUE y FALSE basado en isnan

Al negar dichos valores (~), los valores que son FALSE se vuelven

TRUE, y se corresponden con aquellos valores que NO son nan

Si el vector x, que contiene los valores en el eje x, se afectan

a partir de dichos valores lógicos, se genera un nuevo vector en

el que solos se toman aquellos que son TRUE. Por tanto, se crea

un nuevo vector x, en el cual han desaparecido los correspondientes

valores de y que son nan

Esto mismo se aplica, pero sobre el vector y, lo cual hace que tanto

x como y queden completamente sincronizados: sin valores nan

x = x[~np.isnan(y)]

y = y[~np.isnan(y)]

con el siguiente código, se van a definir las rectas a graficar, dando como parámetros, el valor máximo del eje “y” y el valor minimo del eje x

def plot\_models(x, y, models, fname, mx=None, ymax=None, xmin=None):

''' dibujar datos de entrada '''

Se crea una nueva figura, o activa una existente.

num = identificador, figsize: anchura, altura

plt.figure(num=None, figsize=(8, 6))

Borra el espacio de la figura

plt.clf()

Se crea un gráfico de dispersión de y frente a x con diferentes tamaños y colores de marcador (tamaño = 10)

plt.scatter(x, y, s=10)

Se define el título que van a tener las gráficas y los títulos de los ejes.

plt.ylabel("Contagios")

plt.xlabel("Semanas")

plt.title("Nuevos contagios por día COVID-19")

se implementan etiquetas verticales, para marcar las semanas a lo largo del tiempo:

plt.xticks(

[w \* 7 for w in range(30)],

['%i' % w for w in range(30)])

Se definen los tipos de modelos recibidos, para poder dibujar las curvas de ajuste.

linspace devuelve números espaciados uniformemente

durante un intervalo especificado. En este caso, sobre

el conjunto de valores x establecido

La función zip () toma elementos iterables (puede ser cero o más), los agrega en una tupla y los devuelve

if models:

if mx is None:

mx = np.linspace(0, x[-1], 1000)

for model, style, color in zip(models, linestyles, colors):

plt.plot(mx, model(mx), linestyle=style, linewidth=2, c=color)

el plt.plot lo utilizamos para personalizar el modelado o grafica que esta elaborando.

el método legend de un conjunto de ejes muestra la leyenda en el gráfico. Vimos en su momento que uno de los parámetros que podemos pasar a la función plot es label. Esta etiqueta es la que se mostrará en la leyenda representando a la gráfica

plt.legend(["d=%i" % m.order for m in models], loc="upper left")

plt.legend(["d=%i" % m.order for m in models], loc="upper left")

Activa o desactiva el ajuste de escala automático y luego, si el ajuste de escala automático para cualquiera de los ejes está activado, realiza el ajuste de escala automático en el eje o ejes especificados.

plt.autoscale(tight=True)

grid muestra las líneas de la cuadrícula. Si se proporcionan, se supone que desea que la cuadrícula esté activada

y se establecerá en True

plt.ylim(ymin=0)

if ymax:

plt.ylim(ymax=ymax)

if xmin:

plt.xlim(xmin=xmin)

plt.grid(True, linestyle='-', color='0.75')

el plt.savefig guarda la figura con su respectivo nombre, se guarda en el directorio CHARTS

plt.savefig(fname)

Utiliza la clase plot\_models descrita anteriormente, enviando como parametro el eje x, eje y, None que se refiere al

modelo y envia el archivo en formato png

Crea y dibuja los modelos de datos

plot\_models(x, y, None, os.path.join(CHART\_DIR, "1400\_01\_01.png"))

Se define la función numero 1 con la instruccion np.polyfit es un ajuste polinomial de mínimos cuadrados.

Se ajusta un polinomio de grado deg a los puntos (x, y).

Esta devuelve un vector de coeficientes p que minimiza el error al cuadrado en el orden deg x como una matriz con coordenadas, y como un arreglo, 1 indica el grado del polinomio

fp1, res1, rank1, sv1, rcond1 = np.polyfit(x, y, 1, full=True)

print("Parámetros del modelo fp1: %s" % fp1)

print("Error del modelo fp1:", res1)

f1 = sp.poly1d(fp1)

fp2, res2, rank2, sv2, rcond2 = np.polyfit(x, y, 2, full=True)

print("Parámetros del modelo fp2: %s" % fp2)

print("Error del modelo fp2:", res2)

f2 = sp.poly1d(fp2)

f3 = sp.poly1d(np.polyfit(x, y, 3))

f10 = sp.poly1d(np.polyfit(x, y, 10))

f100 = sp.poly1d(np.polyfit(x, y, 100))

Se grafican los modelos

plot\_models(x, y, [f1], os.path.join(CHART\_DIR, "1400\_01\_02.png"))

plot\_models(x, y, [f1, f2], os.path.join(CHART\_DIR, "1400\_01\_03.png"))

plot\_models(

x, y, [f1, f2, f3, f10, f100], os.path.join(CHART\_DIR,

"1400\_01\_04.png"))

Ajusta y dibuja un modelo utilizando el conocimiento del punto de inflexión

inflexion = 18 \* 7

xa = x[:int(inflexion)]

ya = y[:int(inflexion)]

xb = x[int(inflexion):]

yb = y[int(inflexion):]

Se grafican dos líneas rectas

a = sp.poly1d(np.polyfit(xa, ya, 1))

fb = sp.poly1d(np.polyfit(xb, yb, 1))

Se presenta el modelo basado en el punto de inflexión

plot\_models(x, y, [fa, fb], os.path.join(CHART\_DIR, "1400\_01\_05.png"))

Función de error

def error(f, x, y):

return np.sum((f(x) - y) \*\* 2)

Se imprimen los errores

print("Errores para el conjunto completo de datos:")

for f in [f1, f2, f3, f10, f100]:

print("Error d=%i: %f" % (f.order, error(f, x, y)))

print("Errores solamente después del punto de inflexión")

for f in [f1, f2, f3, f10, f100]:

print("Error d=%i: %f" % (f.order, error(f, xb, yb)))

print("Error de inflexión=%f" % (error(fa, xa, ya) + error(fb, xb, yb)))

Se extrapola de modo que se proyecten respuestas en el futuro

plot\_models(

x, y, [f1, f2, f3, f10, f100],

os.path.join(CHART\_DIR, "1400\_01\_06.png"),

El método en Python une uno o más componentes de ruta de forma inteligente.

Este método concatena varios componentes de ruta con exactamente un separador de directorio

('/') después de cada parte no vacía excepto el último componente de ruta

mx=np.linspace(0 \* 7, 29 \* 7, 100),

Devuelve números espaciados uniformemente durante un intervalo especificado.

y el número de muestras espaciadas uniformemente, calculadas sobre el intervalo

ymax=14000, xmin=0 \* 7)

print("Entrenamiento de datos únicamente despúes del punto de inflexión")

fb1 = fb

fb2 = sp.poly1d(np.polyfit(xb, yb, 2))

fb3 = sp.poly1d(np.polyfit(xb, yb, 3))

fb10 = sp.poly1d(np.polyfit(xb, yb, 10))

fb100 = sp.poly1d(np.polyfit(xb, yb, 100))

print("Errores después del punto de inflexión")

for f in [fb1, fb2, fb3, fb10, fb100]:

print("Error d=%i: %f" % (f.order, error(f, xb, yb)))

Gráficas después del punto de inflexión

plot\_models(

x, y, [fb1, fb2, fb3, fb10, fb100],

os.path.join(CHART\_DIR, "1400\_01\_07.png"),

mx=np.linspace(0 \* 7, 29 \* 7, 100),

ymax=14000, xmin=0 \* 7 )

Separa el entrenamiento de los datos de prueba

frac = 0.3

Indica el número de índices

split\_idx = int(frac \* len(xb))

shuffled = sp.random.permutation(list(range(len(xb))))

El shuffle() método toma una secuencia (lista, cadena o tupla) y reorganiza el orden de los elementos sp.random.permutation Permutar aleatoriamente una secuencia o devolver un rango permutado.

test = sorted(shuffled[:split\_idx])

Una clasificación ascendente simple es muy fácil: simplemente llame a la sorted()función. Devuelve una nueva lista ordenada

train = sorted(shuffled[split\_idx:])

fbt1 = sp.poly1d(np.polyfit(xb[train], yb[train], 1))

fbt2 = sp.poly1d(np.polyfit(xb[train], yb[train], 2))

print("fbt2(x)= \n%s" % fbt2)

print("fbt2(x)-20000= \n%s" % (fbt2-20000))

fbt3 = sp.poly1d(np.polyfit(xb[train], yb[train], 3))

fbt10 = sp.poly1d(np.polyfit(xb[train], yb[train], 10))

fbt100 = sp.poly1d(np.polyfit(xb[train], yb[train], 100))

fb se utiliza para crear la funcion con inflexion, recibe xb con los varlores ordenados y yb con los valores tambien ordenados

Se especifica el orden de dicho polinomio

Encuentra las raíces de una función.

from scipy.optimize import fsolve

Devuelve las raíces de las ecuaciones (no lineales) definidas por una estimación inicial dada.

print(fbt2)

print(fbt2 - 20000)

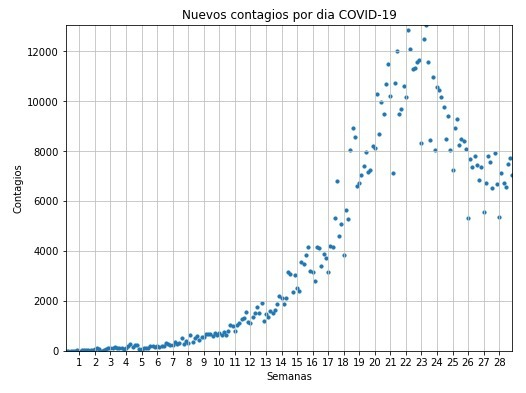
Se resta con 100000 para reducir la escala

alcanzado\_max = fsolve(fbt2 - 20000, x0=203) / (7)

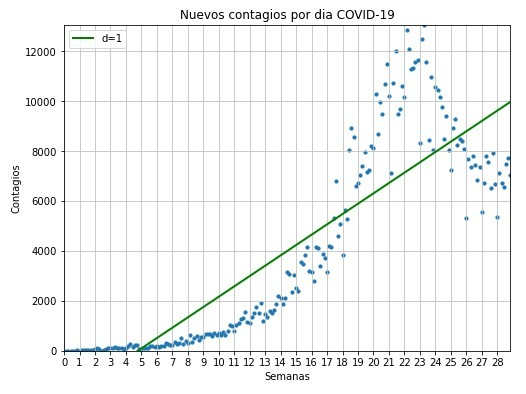
print("\n20000 contagiados esperados en la semana %f" %

alcanzado\_max[0])

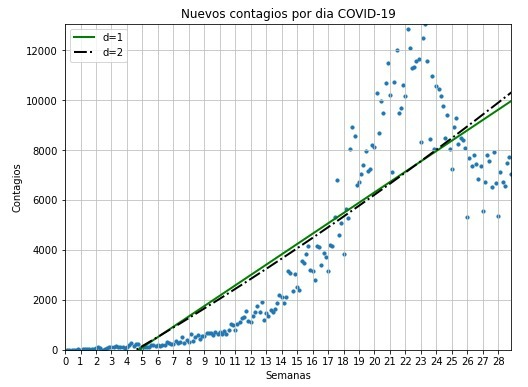
GRAFICAS



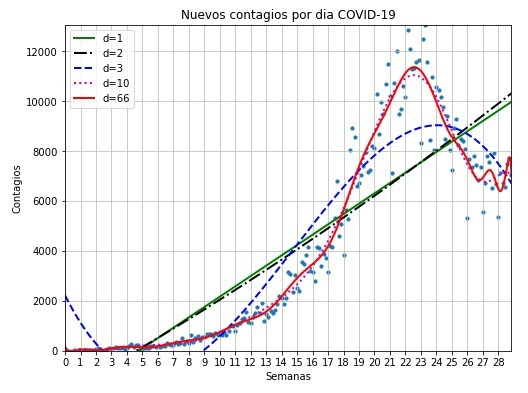
Aquí se puede observar cada uno de los datos en el tiempo, como se representan.



Se hace una regresión lineal de una dimensión, es decir, se traza una recta que trate de pasar por en medio de todos los datos, como podemos ver no es muy precisa, pero cumple con su funcionalidad.



En esta grafica No. 3 se hace de nuevo una regresión lineal, pero esta vez de 2 dimensiones, aunque como podemos ver, no se diferencia mucho de la primera, excepto por una pequeña curvatura.



En esta grafica se implementa una regresión de dimensión 3, la cual se acerca un poco mas al comportamiento de los datos a través del tiempo, también se implementan otras 2, de dimensión 10 y 66, en las que se puede apreciar con mayor exactitud el comportamiento del virus en cuanto a contagios nuevos diarios se refiere.

CONCLUSIONES

Como podemos observar en la última gráfica, el crecimiento de contagios tiene una tendencia exponencial, al menos hasta la semana 22 aproximadamente, donde luego empezará a decaer esta cifra hasta la semana 26, en la que el numero de contagiados diarios se mantendrá en un rango entre 6000 y 8000 contagiados por día.

Podemos notar, que entre mas dimensiones tenga una regresión lineal, mas acertada y cercana estará al comportamiento de los datos.

El valor máximo de contagios es un poco mayor a 12mil, se encuentra entre la semana 23 y 24, sin embargo, este no hace parte de la curva que representa el comportamiento, esto pasará con todos los datos que se alejen de la media a medida que avanza la gráfica, como se puede observar con otros datos dispersos y lejanos a ella.

A medida que pasa el tiempo, entre más días, más dispersión de datos, esto ocurre debido a que la cantidad de contagios alcanza números grandes, por lo que el rango de diferencia aumenta también.

REFERENCIAS

Referencias en la Web:

[1]

<https://datosabiertos.esri.co/datasets/782122624f364fbdbd7e287b96c4a358_6/data?orderBy=FECHA_ACTUALIZACION&selectedAttribute=FECHA_ACTUALIZACION>