

Cálculo de correlaciones temporales entre series temporales de COVID-19

Denise Stefania Cammarota
Introducción al cálculo numérico en Procesadores Gráficos
Instituto Balseiro
2 de julio de 2021
denisescammarota@gmail.com

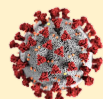


Resumen de la charla

1. Introducción al problema, objetivos
2. Procesamiento de datos y cálculo de correlaciones
3. Comparación de performance
4. Resultados y análisis
5. Conclusiones

Introducción y objetivos

Introducción: COVID-19



SARS-COV-2 → COVID-19

(Severe Acute Respiratory
Syndrome Coronavirus 2)

(Coronavirus
Disease 2019)

Síntomas de la COVID-19

- Fiebre, tos seca, fatiga
- Pérdida del olfato y del gusto, dolor de garganta
- **Graves:** falta de aire, pérdida del apetito, dolor en el tórax

Breve línea de tiempo

31/Dic/19: Primeros casos en Wuhan

9/Enero/20: Identificación del virus

30/Enero/20: Alerta de la OMS

1/Mar/20: 1er caso importado Argentina

11/Mar/20: OMS declara pandemia

20/Mar/20: Comienzo ASPO

...

2021: continua el estado de pandemia

Situación actual

Argentina (30/jun/21)

4.470.374

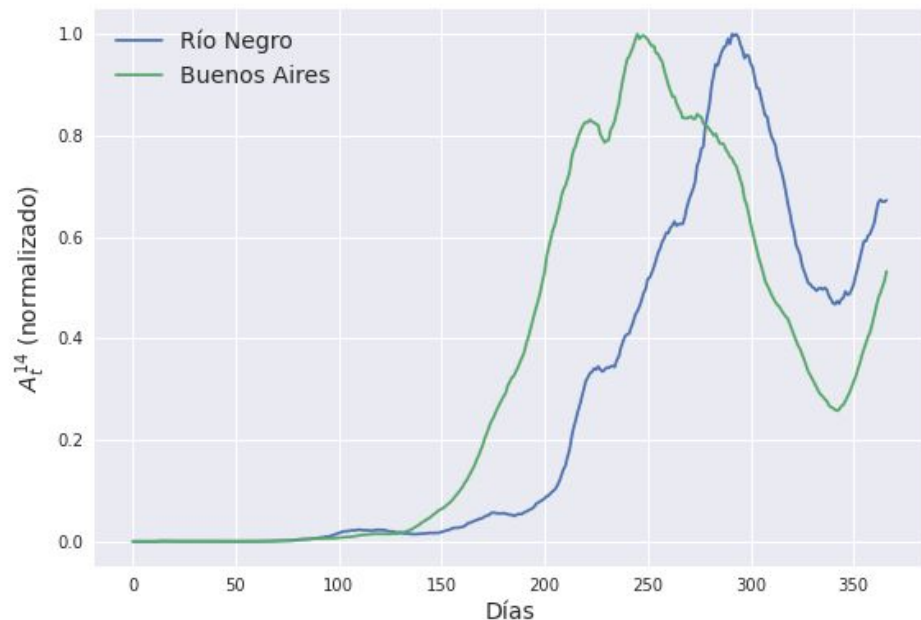
TOTAL DE AFECTADOS

299.149

INFECTADOS ACTIVOS

94.304

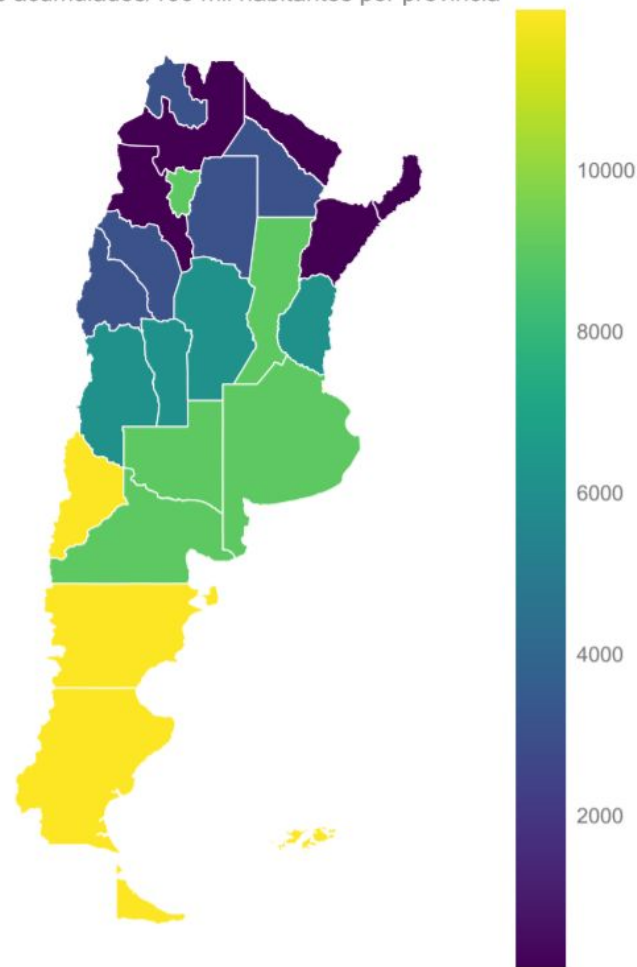
FALLECIDOS



Objetivos

- Analizar la correlación temporal entre series temporales de provincias y localidades
- Observar propiedades: ¿qué localidades tienen mayor correlación entre sí? ¿Cuáles se adelantan a otras? ¿Qué significa esto?
- Ver si esto puede optimizarse utilizando GPU

Casos acumulados/100 mil habitantes por provincia



Datos del Ministerio de Salud

COVID-19. Casos registrados en la República Argentina

Dirección Nacional de Epidemiología y Análisis de Situación de Salud 

<http://datos.salud.gob.ar/dataset/covid-19-casos-registrados-en-la-republica-argentina>



Ministerio de Salud
Argentina



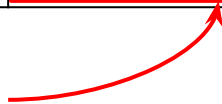
Número de infectados confirmados
en el día t

↓
$$N_t$$

con t día **inicio de síntomas**

| id | residencia_provincia_nombre | residencia_localidad_nombre | fecha_inicio_sintomas | fecha_apertura | clasificación |
|----|-----------------------------|-----------------------------|-----------------------|----------------|---------------|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

puede ser inexistente y hay que tenerlo en cuenta al
procesar los datos




Procesamiento de datos y cálculo de correlaciones

Primer paso: Carga de los datos

Consiste en cargar los datos del ministerio de salud previo al procesamiento. Para eso, utilizamos **cudf** para GPU y **pandas** para CPU. Usamos datos hasta el 1/enero/21, archivo del 11/enero/21.

```
import pandas as pd
data = pd.read_csv(direccion, usecols=columnas, ...)
data = data[data["clasificacion_resumen"] == "Confirmado"]
df = pd.DataFrame(data)
df = df.drop(['clasificacion_resumen'], axis=1)
df["fecha_inicio_sintomas"] -= inicio_epidemia
df["fecha_apertura"] -= inicio_epidemia
df.fecha_inicio_sintomas = df.fecha_inicio_sintomas.dt.days
df.fecha_apertura = df.fecha_apertura.dt.days
```

```
import cudf
data = cudf.read_csv(direccion, usecols=columnas, ...)
data = data[data["clasificacion_resumen"] == "Confirmado"]
df = cudf.DataFrame(data)
df = df.drop(['clasificacion_resumen'], axis=1)
df["fecha_inicio_sintomas"] -= inicio_epidemia
df["fecha_apertura"] -= inicio_epidemia
df.fecha_inicio_sintomas = df.fecha_inicio_sintomas.dt.days
df.fecha_apertura = df.fecha_apertura.dt.days
```




Segundo paso: Pre-procesamiento de los datos

Consiste en un pre-procesamiento de los datos para modificar las fechas de inicio de síntomas, utilizando **cupy** y **cudf** para GPU, y **numpy** y **pandas** para CPU

```
import numpy as np
df = change_dates(df)
def change_dates(df):
    filt_df2 = (df.fecha_inicio_sintomas.isnull())
    b = np.array(df["fecha_apertura"].values)
    a = np.array(df["fecha_inicio_sintomas"].values)
    a[filt_df2.values] = b[filt_df2.values]
                        - np.random.randint(0,9,a[filt_df2.values].shape)
    df["fecha_inicio_sintomas"] = a
```

```
import cupy as cp
df = change_dates(df)
def change_dates(df):
    filt_df2 = (df.fecha_inicio_sintomas.isnull())
    b = cp.array(df["fecha_apertura"].values)
    a = cp.array(df["fecha_inicio_sintomas"].values)
    a[filt_df2.values] = b[filt_df2.values]
                        - cp.random.randint(0,9,a[filt_df2.values].shape)
    df["fecha_inicio_sintomas"] = a
```



Tercer paso: Obtención de series temporales

Consiste en agrupar por provincia/localidad y fecha de inicio de síntomas. A partir de esto, podemos obtener las series temporales para todas las provincias/localidades.

```
df = df[["residencia_provincia_nombre", "fecha_inicio_sintomas"]]
df2 = df.groupby(["residencia_provincia_nombre", "fecha_inicio_sintomas"])
df2 = df2.rename(columns={0: 'casos'})
provincias = cudf.Series(df3["residencia_provincia_nombre"].unique())
rows = provincias.shape[0]
cols = df2["fecha_inicio_sintomas"].max()
time_series = cp.zeros(shape=(rows, cols+1))
j = 0
for i in provincias:
    df_tmp = df3[df3["residencia_provincia_nombre"] == i]
    tmp_dates = cp.array(df_tmp["fecha_inicio_sintomas"].values)
    tmp_cases = cp.array(df_tmp["casos"].values)
    time_series[j, tmp_dates] = tmp_cases
    j = j + 1
time_series_ac = cp.zeros(shape=(rows, cols+1))
#suma para obtener acumulados en 14 días
...
```

Provincias/
Localidades

Tiempo en días

| | | | |
|-----|-----|-----|-----|
| 1 | 10 | 24 | 50 |
| ... | ... | ... | ... |
| 0 | 1 | 3 | ... |

Cuarto paso: Cálculo de correlaciones y lags óptimos

Finalmente, calculamos las correlaciones y lags óptimos para cada par de provincias/localidades. Utilizamos las funciones **cupy.correlate** de cupy, y **numpy.correlate** de numpy. Guardamos los valores en una matriz.

```
while i < rows:
    prov_tmp_1 = time_series_ac[i, :-10]
    while j < rows:
        prov_tmp_2 = time_series_ac[j, :-10]
        ccov = cp.correlate(prov[i] - prov[i].mean(),
        prov[j] - prov[j].mean(), ...)
        ccor = ccov / (npts * prov[i] * [j])
        localidades_lagmax[i, j] = lags[np.argmax(ccor)]
        localidades_corrmax[i, j] = np.max(ccor)
```

Matriz tal que:

$[i, j]$ = correlación localidad i con j

Matriz tal que:

$[i, j]$ = lag localidad i con j

lag < 0: i antecede a j

lag > 0: i precede a j

Comparación de performance

Comparación de la performance: GPU vs CPU

GPU

NVIDIA GeForce GTX 1650, 4GB RAM

Tiempo total

Provincias: (8 ± 3) s

Localidades: ~ 1200 s

Carga de datos

Provincias: (2 ± 2) s

Localidades: ~ 15 s

Cálculo de correlaciones

Provincias: (3.5 ± 0.6) s

Localidades: ~ 1200 s

CPU

Intel Core i5 9300-H @ 2.40GHz

Tiempo total

Provincias: (10.4 ± 0.4) s

Localidades: (37 ± 3) s

Carga de datos

Provincias: (9.8 ± 0.4) s

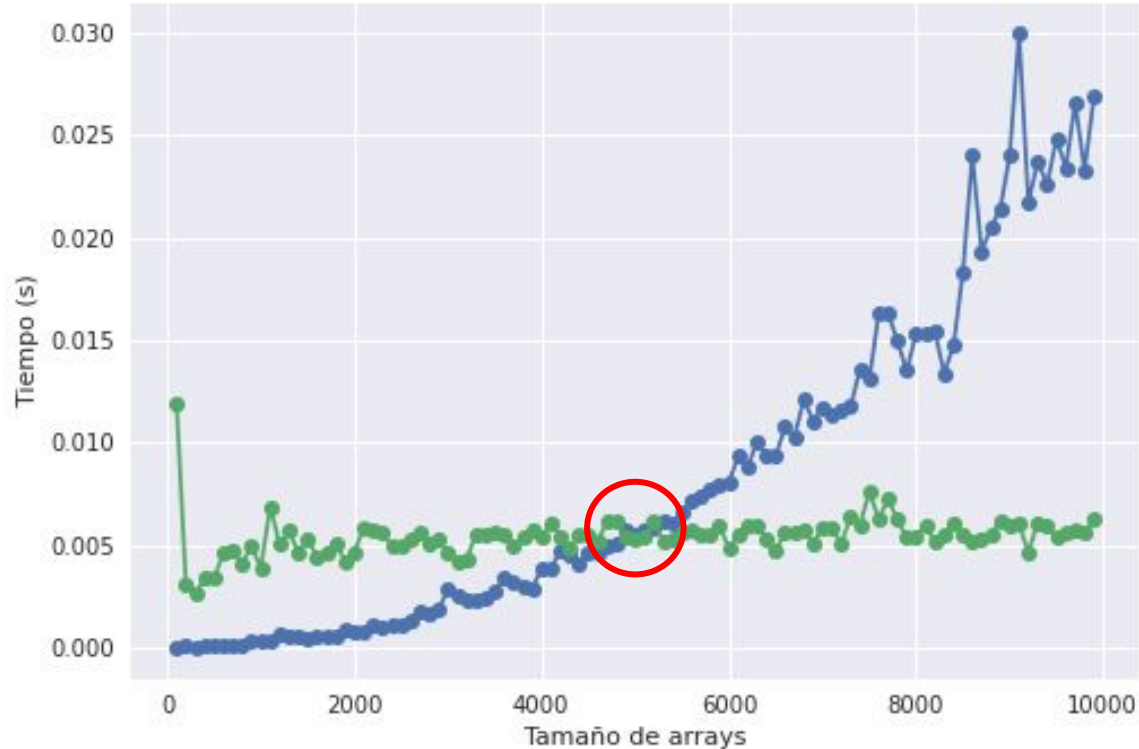
Localidades: (10 ± 2) s

Cálculo de correlaciones

Provincias: (0.09 ± 0.01) s

Localidades: (23 ± 2) s

Análisis de performance

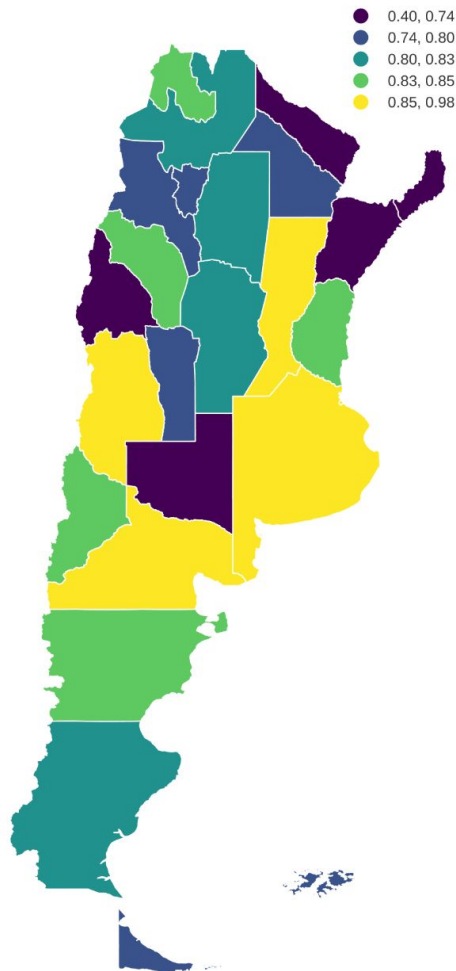


- Generamos dos vectores aleatorios de un tamaño
- Calculamos la correlación con `np.correlate` y `cp.correlate`
- Calculamos el tiempo que toma cada operación para distintos tamaños

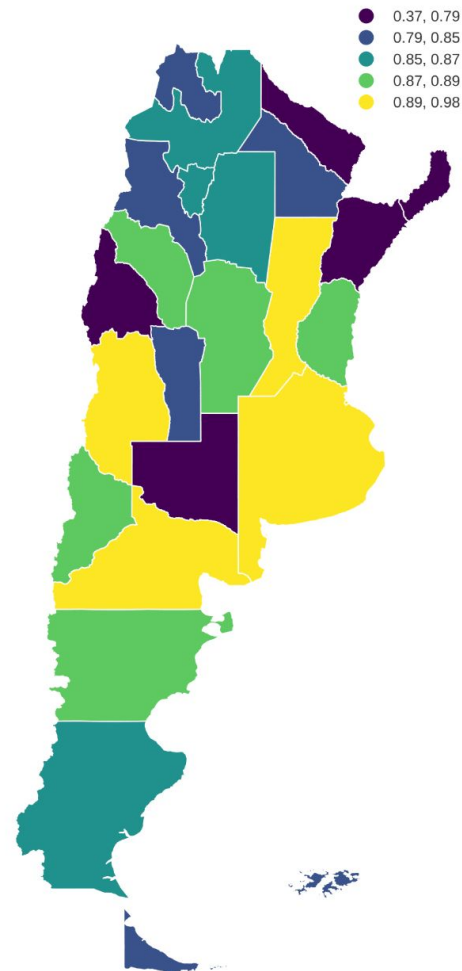
Resultados y Análisis

Mapas de correlación: CABA y Provincia de Buenos Aires

Correlación con CABA

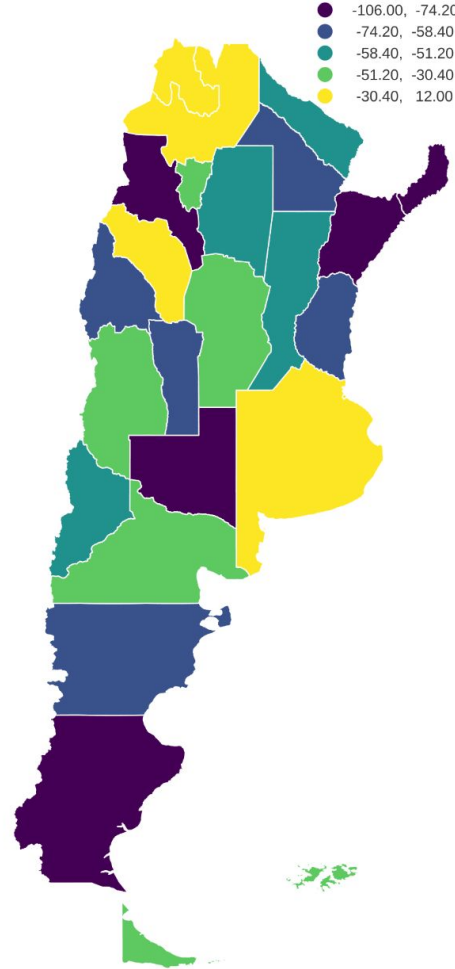


Correlación con Provincia de Buenos Aires

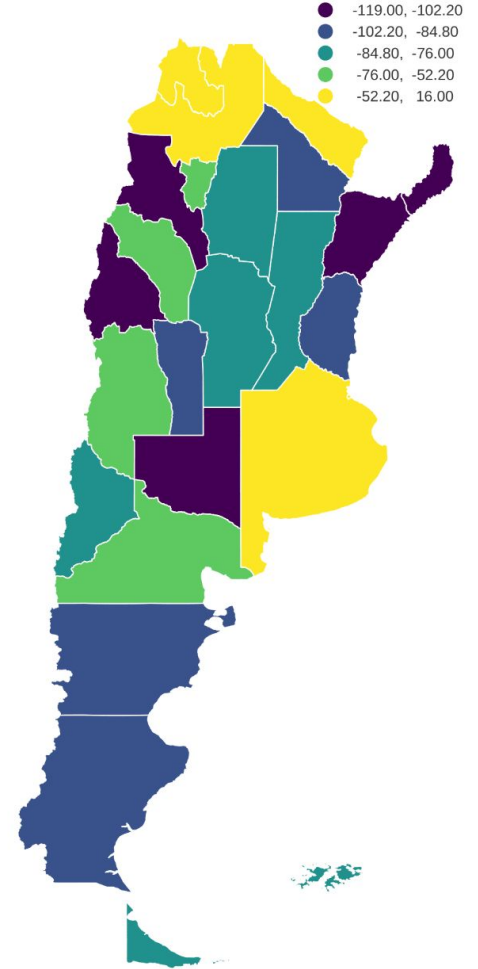


Mapas de lags: CABA y Provincia de Buenos Aires

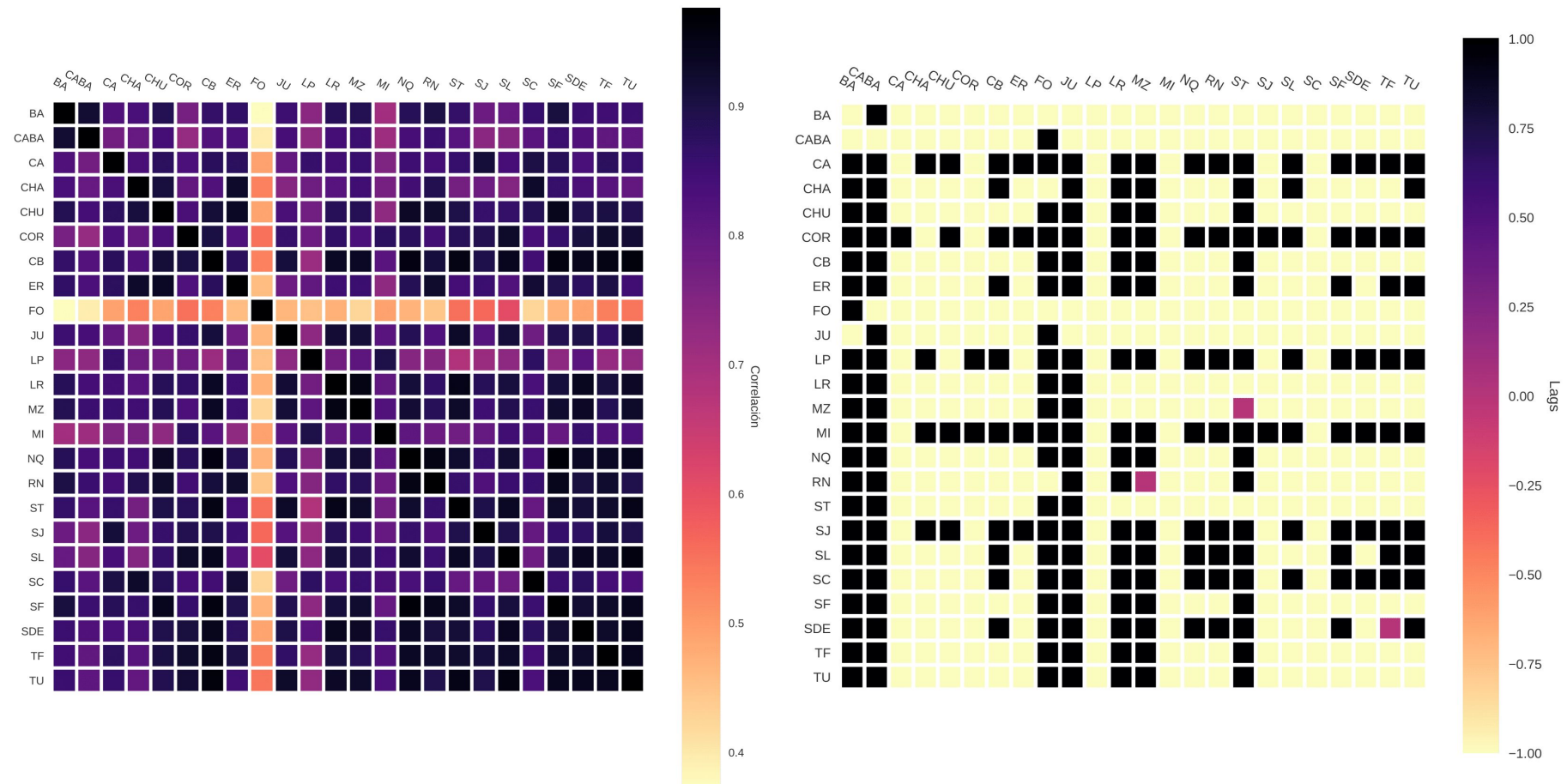
Lag con Provincia de Buenos Aires



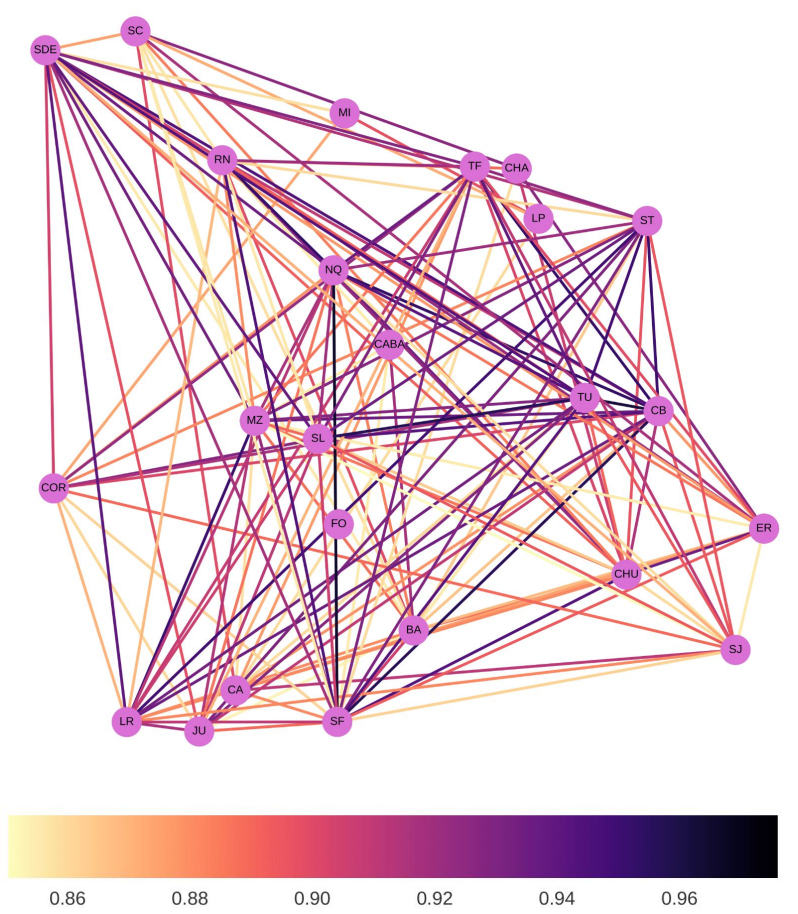
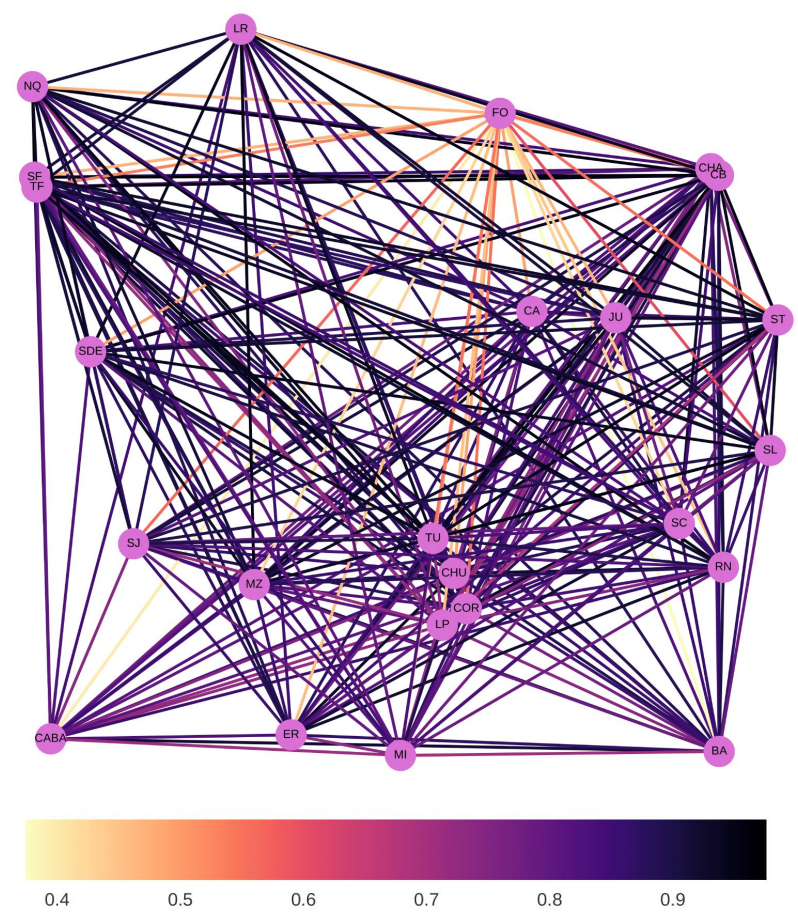
Lag con CABA



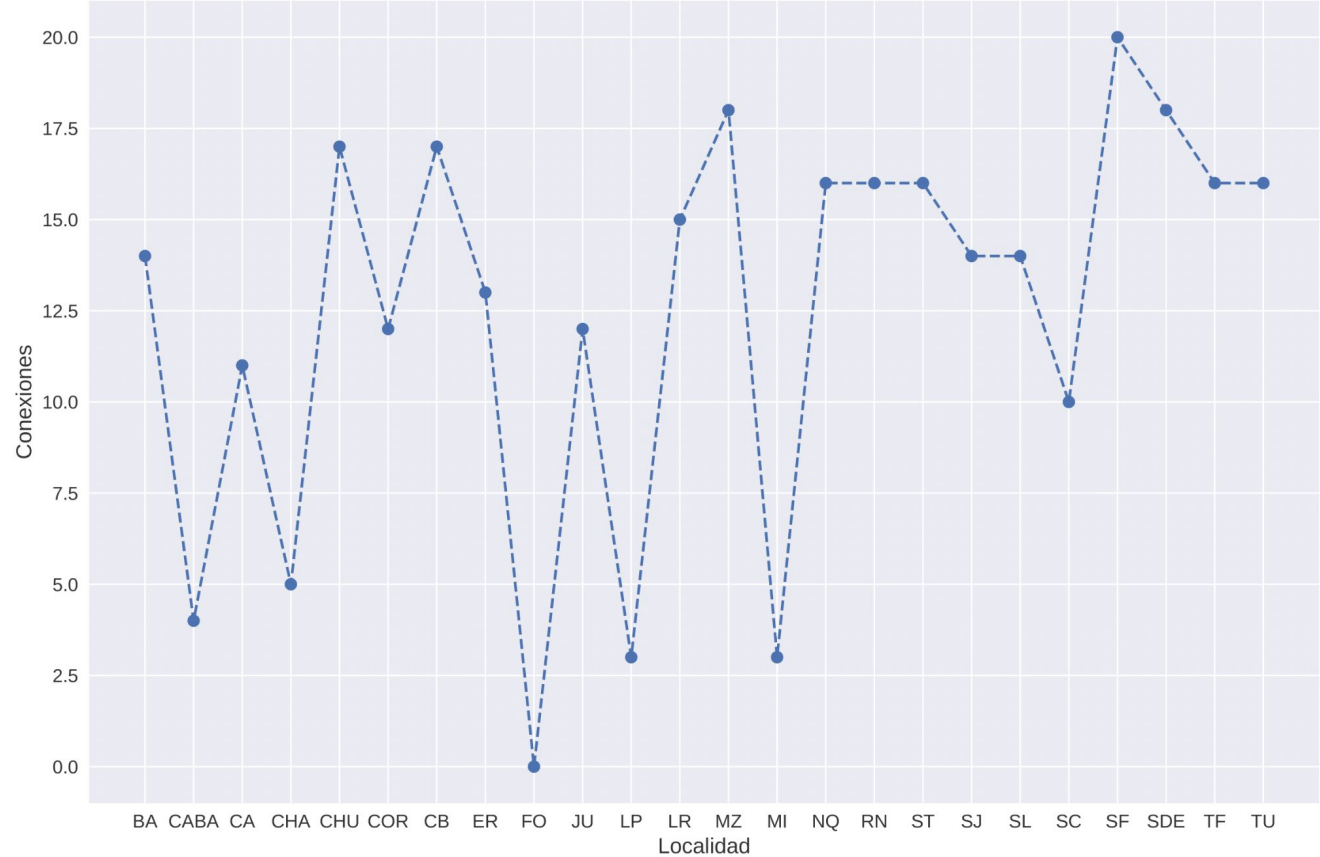
Matrices de correlación y de lag entre provincias



Redes dadas por la correlación



Conexiones para cada provincia



Conclusiones

Conclusiones

- Se realizaron programas en serie y paralelo para calcular correlaciones temporales entre series de casos de COVID-19
- Para ello, se utilizaron en GPU las librerías cudf y cupy de Python
- Se compararon los rendimientos de ambos programas
- Se pudieron observar propiedades de las correlaciones temporales
- Como trabajo a futuro, queda profundizar en el análisis de los resultados y investigar otros métodos para acelerar el cálculo

¡Muchas gracias!

¿Preguntas?