

# Reporte de Seaborn

Eduardo Peñuñuri Bolado

Grupo 1

17 de marzo de 2019

## Reporte

En la actividad realizada, hicimos básicamente gráficos pero utilizando, además de Matplotlib, Seaborn, para poder comparar los gráficos, facilidad de usarlos, etc., y así poder ver cual es mejor o más eficiente y/o sencillo de usar, además de que hicimos otros gráficos con los datos trabajados. Cabe aclarar, que para ello, a parte de invocar Pandas, se hizo lo mismo con Matplotlib y Seaborn.

Lo primero que hice fue trabajar los datos que se nos proporcionaron, ya que contaba con algunos espacios vacíos o regiones innecesarias, que podrían hacer que los resultados finales se vieran perjudicados negativamente, generando errores importantes que luego serían difíciles de erradicar. Para ello, después de leer los datos en Python, ejecutamos el siguiente comando para eliminar huecos sin información de las tablas:

```
#Eliminamos las columnas que no contienen datos
df1 = df0.drop(df0.columns[df0.columns.str.contains('unnamed: ', case=False)], axis=1)
#df1
```

Posteriormente, revisamos el tipo de variable de las columnas (haciendo uso de .dtypes), y filtramos los datos, para quedarnos con los de 2019:

```
#Eliminamos los datos que sean de 2010
df1.drop(range(52338, 52639), axis=0)
df1.head()
```

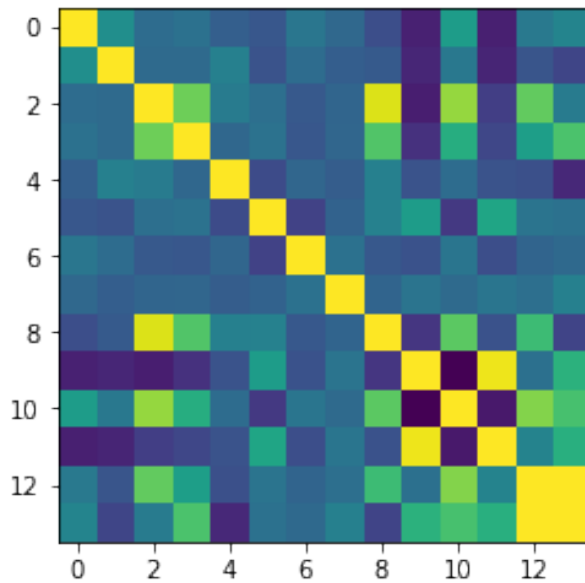
Y creamos otra columna para la fecha, para posteriormente volver a checar los tipos de variables y ver que todo se encuentre en orden:

```
#Creamos una columna Fecha para la fecha y quitamos la columna de DATE
#Primero hacemos otro DataFrame pero sin el primer renglón (el de las unidades),
para luego trabajar con ella
dfA = df1
dfA = df1.drop(0, 0)
```

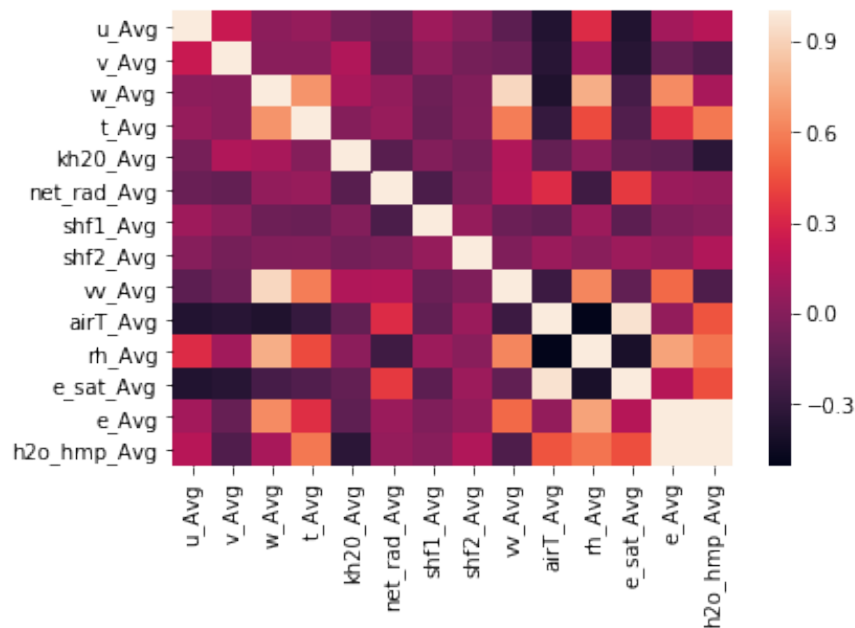
```
#Creamos una columna Fecha para la fecha y quitamos la columna de DATE
dfA["FECHA"] = pd.to_datetime(dfA.apply(lambda x:x["DATE"],1),dayfirst=True)
df2 = dfA.drop(["DATE"],1)
df2.head()
```

Lo que siguió en la actividad fue mostrar la correlación, y fue tan sencillo como utilizar la función `corr()`, y una vez obtenido esto, hicimos una gráfica o mapa de correlaciones utilizando Matplotlib y Seaborn, dando lo siguiente:

Matplotlib



Seaborn

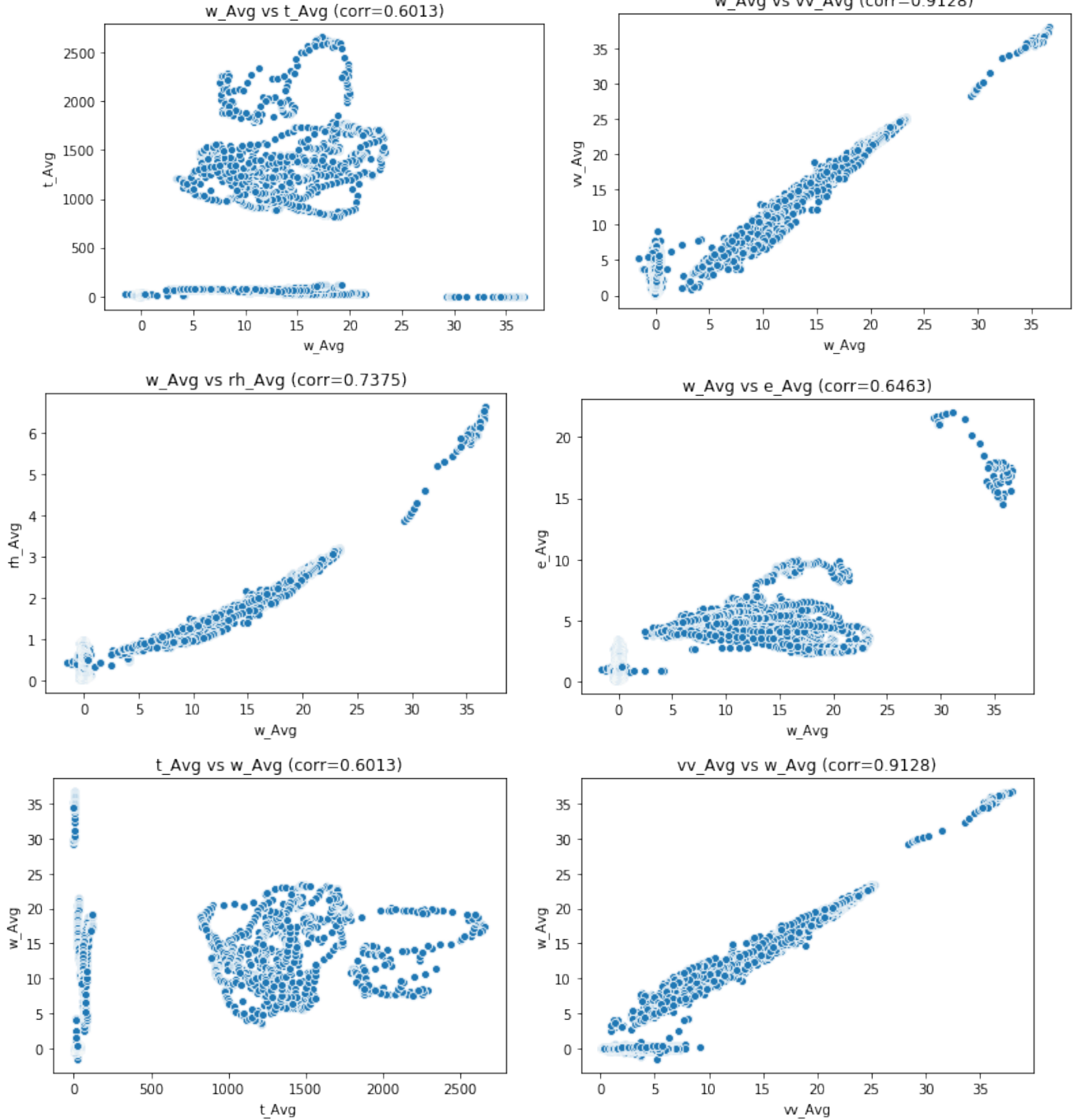


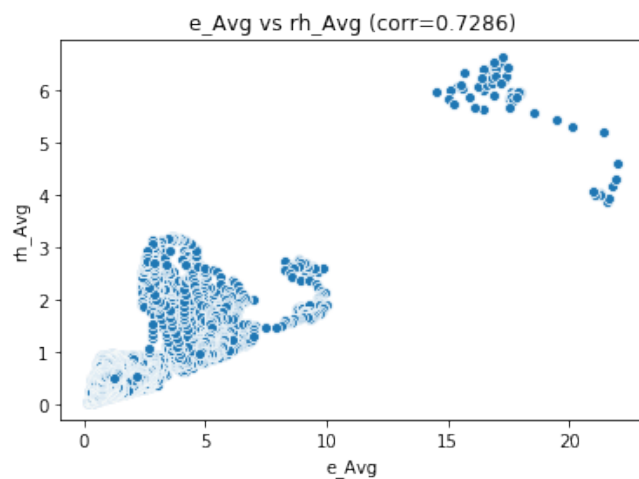
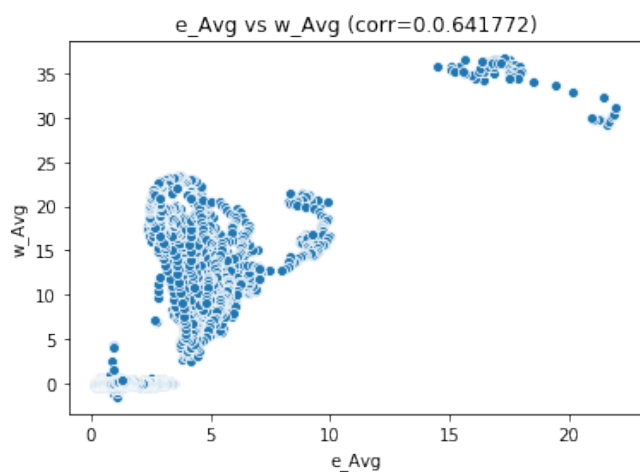
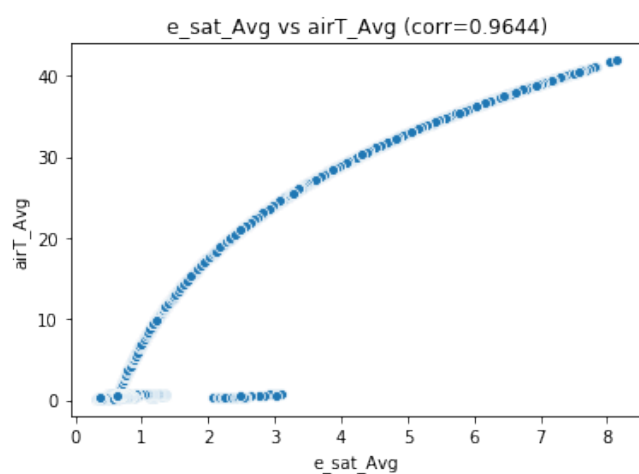
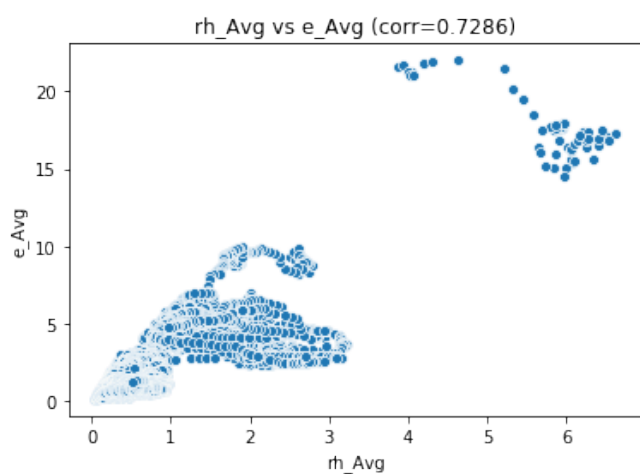
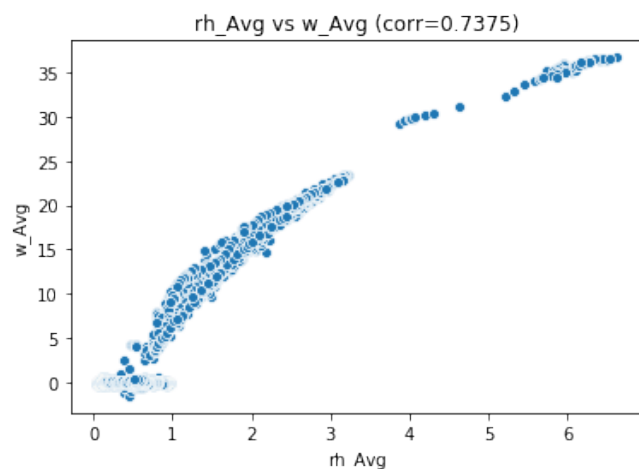
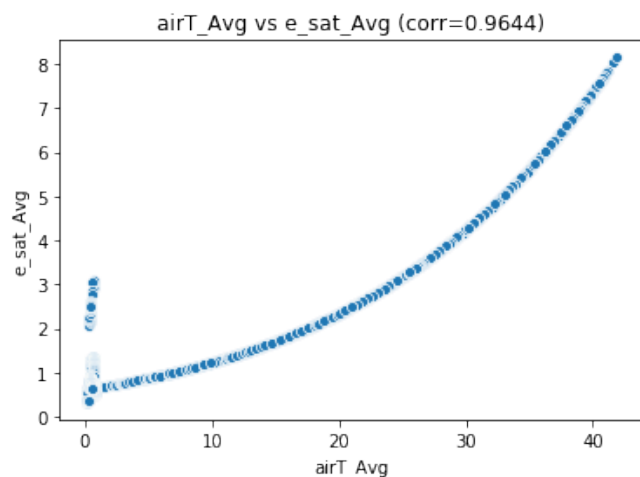
Por último, hicimos gráficas de dispersión de puntos de “Variable 1 vs. Variable 2”, para

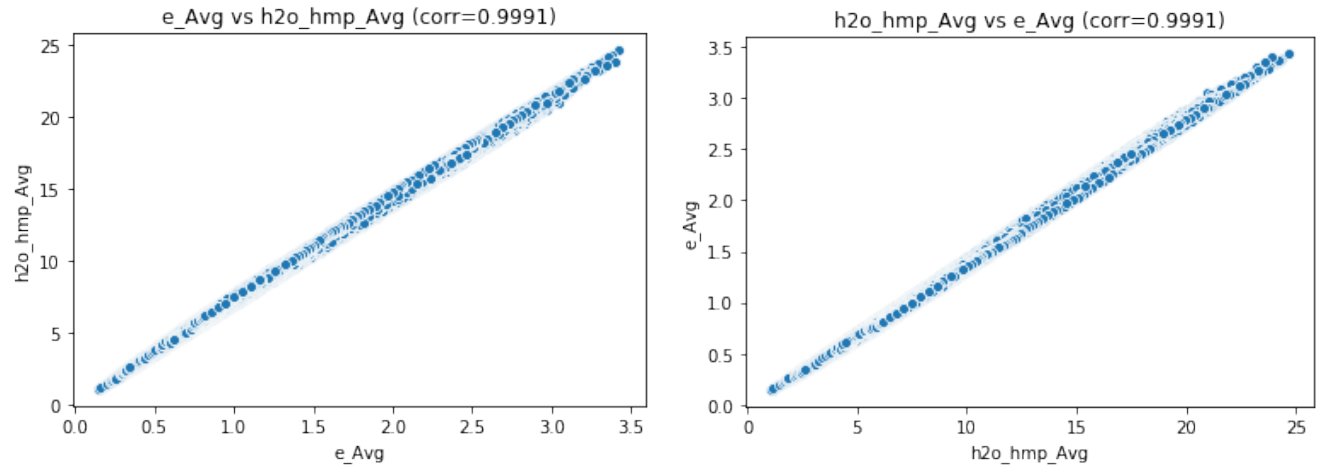
los casos donde la correlación  $\text{abs}(\text{corr})$  mayor a 6, siendo estos casos:

$(w_{Avg}, t_{Avg})$ ,  $(w_{Avg}, vv_{Avg})$ ,  $(w_{Avg}, rh_{Avg})$ ,  $(w_{Avg}, e_{Avg})$ ,  $(t_{Avg}, w_{Avg})$ ,  $(vv_{Avg}, w_{Avg})$ ,  
 $(airT_{Avg}, e_{sat_{Avg}})$ ,  $(rh_{Avg}, w_{Avg})$ ,  $(rh_{Avg}, e_{Avg})$ ,  $(e_{sat_{Avg}}, airT_{Avg})$ ,  $(e_{Avg}, w_{Avg})$ ,  
 $(e_{Avg}, rh_{Avg})$ ,  $(e_{Avg}, h2o_{hmp_{Avg}})$ ,  $(h2o_{hmp_{Avg}}, e_{Avg})$ .

El resultado es las gráficas que se mostrarán a continuación:







## Conclusión

La actividad mostró otra forma de graficar a parte de Matplotlib, la cual es bastante útil y variada, aunque para ser sincero no encontré mucha diferencia en cuanto a la dificultad para usarla, pero si en los resultados, ya que considero que son más completas o presentables con Seaborn que con Matplotlib, sin necesidad de agregar mucho código, mientras que con el otro si queríamos columnas con nombres o cosas similares requería un esfuerzo extra, por lo que Seaborn entrega mejores resultados con menos esfuerzo o dificultad, pero fuera de eso, ambos son similares y pueden utilizarse sin problema, dando una buena variedad de opciones si se sabe buscar.

## Referencias

[1] Michael Waskom (2012). *Seaborn: Statistical data visualization*. Recuperado en marzo de 2019 de: <https://seaborn.pydata.org/>

[2] Matplotlib development team (2012). *Matplotlib*. Recuperado en marzo de 2019 de: <https://matplotlib.org/>

[3] The python graph gallery (2017). *Seaborn*. Recuperado en marzo de 2019 de: <https://python-graph-gallery.com/seaborn/>